



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

1984

An investigation into the use of data bases in computer-aided naval ship design

CeLotto, Richard Charles

Massachusetts Institute of Technology

<http://hdl.handle.net/10945/19404>

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 94064

DUDLEY KNOX LIBRARY
HARVEY POSTGRADUATE SCHOOL
ROCKFORD, ILL. 61080

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN INVESTIGATION INTO THE USE OF DATA BASES IN COMPUTER-AIDED NAVAL SHIP DESIGN (VOL I&II)		5. TYPE OF REPORT & PERIOD COVERED THESIS
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) CELOTTO, RICHARD C.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS MASS. INST. OF TECHNOLOGY CAMBRIDGE, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS CODE 031 NAVAL POSTGRADUATE SCHOOL MONTEREY, CA 93940		12. REPORT DATE JUN 81
		13. NUMBER OF PAGES 266
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLAS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) NAVAL SHIP DESIGN COMPUTER-AIDED SHIP DESIGN		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) ATTACHED		

T202030

ABSTRACT

A design-oriented, interactive computer system which makes possible the dynamic loading of programs at the user's request throughout the operating session has been developed. This system, which is referred to as DEX, also allows the user to select various types of files as the source and destination of information during the session. With respect to one type of file, databases, DEX introduces a more versatile form of organization and use.

An extended DEX library of subroutines is developed which enables the user to read and write integer scalar, real scalar and one-dimensional real array variables and to edit from the terminal integer and real scalar values. It also enables the user to employ during input and output sequences the unit system of his choice.

A proposal is offered for the organization of DEX databases for the preliminary design of naval ships. Suggestions are made, based on a demonstration computer program, for employing existing ship databases to support a generalized ship synthesis model.

Approved for public release;
distribution unlimited.

AN INVESTIGATION
INTO THE USE OF DATA BASES
IN COMPUTER-AIDED NAVAL SHIP DESIGN

by

RICHARD CHARLES CELOTTO

Lieutenant, United States Navy
B.S., Webb Institute of Naval Architecture
and Marine Engineering
(1973)

SUBMITTED TO THE DEPARTMENT OF
OCEAN ENGINEERING
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS OF THE DEGREES OF

OCEAN ENGINEER

and

MASTER OF SCIENCE IN
NAVAL ARCHITECTURE AND MARINE ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1981

© Richard Charles Celotto 1981

The author hereby grants to M.I.T. permission to reproduce and
to distribute copies of this thesis document in whole or in part.

thesis

C 338367

81

AN INVESTIGATION
INTO THE USE OF DATA BASES
IN COMPUTER-AIDED NAVAL SHIP DESIGN

by

RICHARD CHARLES CELOTTO

Submitted to the Department of Ocean Engineering
on May 8, 1981 in partial fulfillment of
the requirements for the degrees of
Master of Science in
Naval Architecture and Marine Engineering
and
Ocean Engineer

ABSTRACT

A design-oriented, interactive computer system which makes possible the dynamic loading of programs at the user's request throughout the operating session has been developed. This system, which is referred to as DEX, also allows the user to select various types of files as the source and destination of information during the session. With respect to one type of file, databases, DEX introduces a more versatile form of organization and use.

An extended DEX library of subroutines is developed which enables the user to read and write integer scalar, real scalar and one-dimensional real array variables and to edit from the terminal integer and real scalar values. It also enables the user to employ during input and output sequences the unit system of his choice.

A proposal is offered for the organization of DEX databases for the preliminary design of naval ships. Suggestions are made, based on a demonstration computer program, for employing existing ship databases to support a generalized ship synthesis model.

Thesis Supervisor: Chryssostomos Chryssostomidis
Title: Associate Professor of Ocean Engineering

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to Professor Chryssostomos Chryssostomidis for his guidance, advice and unfailing encouragement which made this thesis possible. The author is especially grateful for the generous sharing of his time throughout the undertaking.

The author would also like to thank his lovely fiancée, Kathy, for her patience and perspective.

TABLE OF CONTENTS

	Page
ABSTRACT	2
ACKNOWLEDGEMENTS	3
TABLE OF CONTENTS	4
LIST OF FIGURES.	8
LIST OF TABLES	9
CHAPTER 1 INTRODUCTION TO DEX.	10
1.1 Background	10
1.2 Description of DEX.	12
1.2.1 Theory	12
1.2.2 Organization	21
1.3 The Extended DEX Library.	25
1.3.1 Environment.	26
1.3.2 Readers	26
1.3.3 Editors	27
1.3.4 Writers	27
1.3.5 Units	28
1.4 DEX Databases	30
1.4.1 Philosophy	30
1.4.2 Format of Database Entries.	30
CHAPTER 2 THE CUBE MODULE SAMPLE PROGRAM	33
2.1 General Description	33
2.1.1 Function of the Module	33
2.1.2 Module Subprograms	33
2.1.3 Typical Operation.	34
2.2 Frequently Used Subroutines.	42
2.2.1 BLOCK DATA	42
2.2.2 MAINPG	45
2.2.3 MODIO.	47
2.2.4 MXUNIT and the "All" Logic.	47
2.3 The Input Series	51
2.3.1 INPUT.	51
2.3.2 DIMENS	51
2.4 The Output Series Subprograms	56
2.5 General Programming Comments	56
CHAPTER 3 THE EXTENDED DEX LIBRARY ENVIRONMENT SETTING ROUTINES.	59

TABLE OF CONTENTS (cont'd)

	Page
3.1 Introduction	59
3.2 Subroutine DIALOG	59
3.2.1 Menu and Calling Parameter	59
3.3 Subroutine SOURCE	62
3.3.1 Menu and Calling Sequence	62
3.3.2 Operation of SOURCE	64
3.4 Subroutine DESTIN	65
3.4.1 Menu and Calling Sequence	65
3.4.2 Operation of DESTIN	65
3.5 Subroutine MDMODE	66
3.6 Subroutine CHKRNG	67
 CHAPTER 4 THE EXTENDED DEX LIBRARY READING ROUTINES	 68
4.1 General Description.	68
4.1.1 Function	68
4.1.2 Organization.	68
4.2 Integer Scalar Series	70
4.2.1 ISCLDR	70
4.2.2 ISCPMP	73
4.2.3 ISREAD	75
4.3 Real Scalar Series	77
4.3.1 Brief Description	77
4.3.2 RSCLDR	77
4.3.3 RVAPMP	79
4.3.4 RSREAD	80
4.4 Real Array Series	82
4.4.1 Brief Description	82
4.4.2 RALLDR.	82
4.4.3 RALRED.	83
 CHAPTER 5 THE EXTENDED DEX LIBRARY EDITING ROUTINES	 87
5.1 General Description	87
5.2 Logical Function ISCEDT	88
5.2.1 Calling Sequence	88
5.2.2 Operation.	90
5.3 Logical Function RSCEDT	91
5.3.1 Calling Parameters.	91
5.3.2 Operation.	91

TABLE OF CONTENTS (cont'd)

	Page
CHAPTER 6 THE EXTENDED DEX LIBRARY WRITING ROUTINES.	93
6.1 General Description.	93
6.2 Integer Scalar Series	95
6.2.1 ISCDMP.	95
6.2.2 ISDSCR	97
6.2.3 ISRITE.	98
6.3 Real Scalar Series	100
6.3.1 RSCDMP.	101
6.3.2 RVDSCR.	102
6.3.3 RSRITE.	103
6.4 Real Array Series	105
6.4.1 RARDMP.	105
6.4.2 RARITE.	108
CHAPTER 7 THE EXTENDED DEX LIBRARY UNIT ROUTINES.	112
7.1 General Description.	112
7.2 The I/O Unit Specifiers	113
7.2.1 General Description	113
7.2.2 Characteristics of a Typical Subroutine	117
7.3 The Basic Unit Series	117
7.3.1 General Description	118
7.3.2 Calling Parameters.	118
7.3.3 Execution.	120
7.4 Derived I/O Unit Series	121
7.4.1 Series Description.	121
7.4.2 UPRESS Calling Parameters	123
7.4.3 UPRESS Operation	124
CHAPTER 8 DEVELOPMENT OF A CRUISER-DESTROYER DATABANK AT M.I.T.	127
8.1 Considerations in Database Design	127
8.1.1 Function	128
8.1.2 Types of Databases.	128
8.2 Organization of the MIT Cruiser-Destroyer Databases	131
8.2.1 General Databases	131
8.2.2 Mass Properties Databases	139
8.3 Independent and Dependent Variables	142
8.3.1 Concept	142

TABLE OF CONTENTS (cont'd)

	Page
8.4 Application of DEX: An Example.	145
8.4.1 Function of MACHWT.	145
8.4.2 List of Subprograms and Menus	146
8.4.3 Description of the Subprograms.	148
8.4.4 Results from the MACHWT Module.	153
8.4.5 Future Developments	154
CHAPTER 9 CONCLUSIONS AND RECOMMENDATIONS	157
REFERENCES.	160
APPENDIX A CUBE MODULE LISTING	162
APPENDIX B UNIT SUBROUTINES ABBREVIATIONS AND CALLING SEQUENCES	201
APPENDIX C SAMPLE GENERAL DATABASE LISTING	205
APPENDIX D GENERAL DATABASE ENTRY CODES	210
APPENDIX E SAMPLE WEIGHT AND VERTICAL CENTER OF GRAVITY DATABASE LISTINGS	219
APPENDIX F MACHWT MODULE LISTING.	226

LIST OF FIGURES

	Page
1-1 Menu "LEN.UNIT"	16
1-2 Two Consecutive Operation Menus.	16
1-3 DEX Menus	23
2-1 Cube Module Menus	35
2-2 Cube Module Subprograms Access Routes.	36
2-3 Sample Cube Module Input Session	33
2-4 Sample Cube Module Output.	42
2-5 Comparison of Module Input/Output Flow	58
3-1 Menu "MOD.ALTR"	60
3-2 Menu "SOURCE".	63
3-3 Menu "DESTINAT"	66
8-1 General Database Variable Relationships	144
8-2 MACHWT Module Menus.	149

LIST OF TABLES

Page

7-1	I/O Unit Specifier Subroutines, Menu Names and Units Available114
7-2	I/O Unit Specifier Calling Parameters116
7-3	Basic Unit Calling Parameters116
7-4	Derived I/O Unit Series.122
7-5	Special Unit Names.126
8-1	General Ship Characteristics Database Variables.133
8-2	Input for MACHWT147
8-3	Sample MACHWT Results.154
B-1	Angle Unit Abbreviations.202
B-2	Force Unit Abbreviations.202
B-3	Length Unit Abbreviations202
B-4	Temperature Unit Abbreviations.203
B-5	Time Unit Abbreviations203
B-6	Calling Sequences of Derived Unit Subroutines.	.204
D-1	Payload Shopping List211
D-2	Supplemental Payload Shopping List217
D-3	Index to Non-Payload Reed Code.218

CHAPTER 1

INTRODUCTION TO DEX

1.1 Background

Significant improvements in the capability and efficiency of computer-aided design systems have been achieved in the past decade by the introduction of interactive computer programs. This development was a major advance in providing more flexibility to the user at the input end of the process. However, too many of the innumerable design programs, and the design systems that incorporate collections of them, suffer from several bothersome problems:

- (i) Less, but still excessive, restrictions on the flexibility of the programs to respond to the individual user's needs.
- (ii) Incompatibility of input and output amongst programs and especially between programs and databases.
- (iii) Excessive training time required for users to learn how to use the programs.
- (iv) Inability to be transported to different facilities.

In 1974, researchers at the Massachusetts Institute of Technology and the University of Michigan felt that an investment in the "front end" of computer-aided design systems could eliminate or reduce these characteristics

and result in design tools of greatly enhanced capability [1], [2], [3]. One of their goals was to develop a system that provides the designer almost as much flexibility at the computer terminal as he/she* has when sitting at a desk with pencil, paper, calculator and imagination at their disposal. The system would be easy to use because of a consistent approach to the interface between the user and the programs. Further, it would incorporate a more intelligent approach to the use of databases. They named this concept "DEX", for Design Executive System.

DEX is a self-contained software package that can be adapted to almost any computer system that supports Fortran. It provides an environment for running task-oriented programs, called "modules". It supports primarily, but not exclusively, interactive programs where there is communication between essentially five "parties": the user, the computer, the computation program, the source of input and the destination of output.

The purpose of the work reported in this thesis was to continue the development of the system at the intermediate level in the DEX hierarchy between what is referred to as "(the) DEX" and the "module". This intermediate

* The use throughout this thesis of the proper pronoun "he" and its derivatives when referring to the programmer, user or designer is for the smoothness of the text and is not to imply any presumption of those persons's sex.

category of subprograms is called the "extended DEX library". (The collection of all the design program modules is called the "DEX library".) The function of the extended DEX library is to enable the user to accomplish the following:

- (i) Establish an environment in which the type of dialogue and the source and destination of information is defined.
- (ii) Specify the system of units to be used for input and output.
- (iii) Read information.
- (iv) Edit information.
- (v) Write information.

This investigator's motivation was to advance the development of an extremely capable tool for the field of ship design, but it should be stressed that DEX can be employed by any discipline involving computer-aided design.

1.2 Description of DEX

1.2.1 Theory. Reference [3] provides an original and excellent description of DEX, but this writer felt that some discussion should be presented here to assist the reader in relating to the information offered in this thesis.

There are five characteristics of DEX which reflect the design philosophy of the system:

- (i) The user is in the design loop.
- (ii) The system allows the design process to be executed in more than one sequence.
- (iii) The system talks with the user in plain English.
- (iv) The system is forgiving.
- (v) The system has multiple capabilities for input and output.

1.2.1.1 The user in the design loop. Design processes are typically iterative ones. This is especially true in the ship design process as vividly illustrated by the ship design spiral. Computer programs allow many and/or complicated calculations to be performed quickly. The faster that the results can be analyzed and a new set of calculations initiated, the better. Even more advantageous is the ability to do only part of the calculations and analyze those results to decide whether or not to proceed. DEX extends the degree of spontaneity characteristic of interactive programs by enabling the dynamic starting of modules of the user's choice, by providing more options for sources of information immediately available to the user and by allowing the user to edit and insert information before it is used in calculations or written to its final destination.

1.2.1.2 Flexibility. The increased flexibility offered by DEX manifests itself in two ways. The first, implemented to allow any computer program acceptable to the operating system to be operated in a system incorporating DEX, is that the degree of involvement with DEX is completely within the prerogative of the module author. The term "module author" was introduced in reference [3] and will be adopted here for consistency. It applies to the person who writes the particular application program and who chooses which DEX services to use. As a minimum, the module author can arrange for the user to issue only the following commands to execute the program:*

- . start main (this activates DEX)
- . begin module2 (this starts the user's program)

There would not really be any point to such an exercise but it can be done by fulfilling only two requirements. First, the name of the file containing the module name (e.g., module2 above) must be introduced in the DEX's

The symbol "." will be used to indicate user-typed commands, "\$" will indicate DEX messages and "" will indicate module messages. These symbols are automatically inserted by DEX.

library file. Secondly, the main program of the module must be a subroutine appearing first in a listing of the module.

The second aspect of the DEX's flexibility is the use of "menus" to provide the user with a wide choice of paths to follow to accomplish his goal. A menu is a list of options (a maximum of twelve per menu is possible) from which the user chooses to either define a value or proceed onto the next step of the operation. Some examples should prove helpful.

Figure 1-1 depicts a typical units menu which illustrates the first type of menu. The user would type in sufficient characters to identify the length unit to be used for input or output, e.g., "foot" (or just "f"). The subprogram which includes this menu would accept the choice, if proper, and return control to the subprogram which called it.

Menus are normally not displayed. The user will likely be aware of the menu options and is simply prompted to make a choice. For this example, one would see:

```
*ENTER AN ITEM FROM MENU - LEN.UNIT
```

However, if the menu choices are unknown, the user can type


```

$      + ----- +
$      +   MENU   +
$      + LEN.UNIT +
$      + ----- +
$  1 + INCH      +
$      + ----- +
$  2 + FOOT      +
$      + ----- +
$  3 + STATMILE  +
$      + ----- +
$  4 + NAUTMILE  +
$      + ----- +
$  5 + MILLIMET  +
$      + ----- +
$  6 + CENTIMET  +
$      + ----- +
$  7 + METER     +
$      + ----- +
$  8 + KILOMET   +
$      + ----- +

```

Figure 1-1. Menu "LEN.UNIT"

```

$      + ----- + ----- +
$      +   MENU   +   MENU   +
$      + MOD.IO   + INPUT    +
$      + ----- + ----- +
$  1 + INPUT      + ALL      +
$      + ----- + ----- +
$  2 + OUTPUT     + UNITS    +
$      + ----- + ----- +
$  3 + DONE       + CA       +
$      + ----- + ----- +
$  4 +             + WATER    +
$      + ----- + ----- +
$  5 +             + HULLFORM +
$      + ----- + ----- +
$  6 +             + SPEEDS   +
$      + ----- + ----- +
$  7 +             + DONE     +
$      + ----- + ----- +

```

Figure 1-2. Two Consecutive Operational Menus


```
.$display menu len.unit
```

to have the menu displayed by the DEX. Note that in this case the user himself types "\$". The word "display" is a selection from menu "DEX.MAIN". After reviewing the menu, by typing

```
.continue
```

he is returned to the prompting message for the length unit menu.

The second type of menu option directs the program to proceed to the next operation. Figure 1-2 shows two successive "operational" menus from a theoretical program that estimates horsepower from the Taylor Standard Series. The user would select item "input" from menu "MOD.IO" in order to pass onto the subprogram containing the menu "INPUT". Any of the items from that menu would pass him onto another subprogram.

There is a subtle difference between the menus of Figure 1-1 and 1-2. Observe that the second shows the item "done" in both menus which is absent from the first. A subprogram with a menu containing "done" returns control to the subprogram which called it only when that entry is selected, whereas for the other, without a

"done", control returns automatically once a selection is made. The latter is used in menus where only one choice would be made at any time.

The user is free to choose any item from a menu that is meaningful to him. There is, therefore, no one predetermined path that must be followed when executing a group of menus. Logic, however, may dictate that one specifies units before reading in water properties.

1.2.1.3 Plain English. The messages and queries to the user provided by DEX have been designed to be as clear as possible. The instructions or responses that the user must supply are natural and logical words that would be used in an oral dialogue. An important aspect of these practices is the uniformity of dialogue encountered by the user under DEX. This reduces the effort required to learn how to operate a new program, which is not an insignificant advantage.

1.2.1.4 Forgiving. By extending the capabilities of the conventional design programs with DEX, the user can accomplish more during a session, but this entails more thinking on his part. The probability that errors will occur is therefore higher.

Even the most experienced user makes mistakes. It may be as simple as depressing the wrong key when typing a menu selection or as improper as supplying an integer

when the program wants a real number. When developing the DEX and extended DEX library routines, and the Machinery Weight Estimating Module of Chapter 8, as many potential errors as possible were anticipated and diagnostic messages, in plain English, were provided. In some cases they advise the user of the error and allow him to try again at that same point. In others, especially where a problem is caused by a programming error, control is returned to the user several sequential subroutines prior to where the error occurred, with a message issued to assist in determining where to search for the mistake.

1.2.1.5 Input/Output Capability. DEX enables the communication of information by the dynamic allocation of databases and files, which are the only two storage locations it recognizes. In practice we distinguish between two types of files, such that the list of locations is as follows:

- (i) databases
- (ii) input locations (which are the terminal for alphanumeric characters and a graphics screen for x-y coordinates) and output locations (the terminal screen in the form of menus)
- (iii) disk files.

Now, for the ease of understanding of the user, the environment in which he operates is described as

having a total of five "sources" of information and four "destinations". The term "information" is preferred here to "input" and "output" to preclude a limiting misconception by the reader. The tendency to think of input as data read and output as answers written should be avoided. In fact, the user may need to "write" input to the terminal for inspection, or "read" an output value from the terminal in order to "write" it to a database. For this reason this writer will often apply the term "information" to both input and output variables as values that have a source or destination.

The sources and destinations provided for in the operating environment of the DEX system described in this thesis are listed here:

- (i) DEX-created databases
- (ii) the terminal using DEX routines to read or write alphanumeric characters
- (iii) the terminal or a plotter using graphics routines to read or create x-y plots
- (iv) sequential files
- (v) module default data (source only)

The third capability does not yet exist in the present version of DEX at MIT because all the necessary enabling routines have not yet been implemented. If the user tries to employ it, error messages advise him of this situation.

The user is not confined to using the same type of destination as source, or the same source for all the information of a program. He may read information from one or more databases, for example, and write it to another, or to the terminal or to a file, or all three in succession. The only restriction is that the module can be pointed toward only one source or destination at one time.

1.2.2 Organization. The hierarchy of DEX consists of three levels of programs: the DEX, the extended DEX library and the module. The first two categories comprise a permanent, portable set of subprograms which provides an interface between the computer and the user-supplied module.

1.2.2.1 DEX. The category called DEX consists of several hundred subroutines, each with a very specific function, which provide the foundation, or "umbrella" if you will, for the DEX System. The employment of these subroutines by the module authors results in a uniform appearance of the system to the user of the various modules exercised. The DEX services provide input/output and data utilities and, eventually at MIT, graphics utilities for the module authors. Although the module author and user need not be aware of most of these subprograms, in two areas they draw directly on the features of routines in this category.

The first area, of interest to the module author, is a set of 37 subroutines and functions which the programmer may incorporate into his module to perform certain tasks. References [4] and [5] describe these subprograms and how they are used. Briefly, they are grouped into five categories: control of execution, input, output, database management and character manipulation. Subsequent chapters will refer to these as "DEX routines".

The second area of overt involvement with the DEX category is encountered by the user during operation of a module. When the command

```
.start main
```

is issued, the user enters the DEX environment and is presented with a prompting message for menu "DEX.MAIN". There are six menus at this level of DEX, and these are listed in Figure 1-3. The first instruction given to the user is:

```
$ENTER AN ITEM FROM MENU - DEX.MAIN
```

These items are listed in the rightmost column in the table. Note menu "DEX.DISP" which allows the user to display any menu by typing

```
.display menu menuname
```


\$	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+
\$	+	MENU	+	MENU	+	MENU	+	MENU	+	MENU	+	MENU	+
\$	+	DB-TYPES	+	DBEDCMDS	+	DXYES-NO	+	DEX.ALTR	+	DEX.DISP	+	DEX.MAIN	+
\$	+	-----	+	-----	+	-----	+	-----	+	-----	+	-----	+
\$	1	+	INTEGER	+	CREATE	+	YES	+	TERSE	+	MENU	+	LIBRARY
\$	2	+	REAL	+	STORE	+	NO	+	VERBOSE	+	NEWS	+	HELP
\$	3	+	ARRAY-RL	+	DELETE	+		+	KEYBOARD	+	MODE	+	DISPLAY
\$	4	+		+	COMMENT	+		+	GRAPHIC	+		+	ALTER
\$	5	+		+	EXPLAIN	+		+	ECHO-ON	+		+	TIDY
\$	6	+		+	PRINT	+		+	ECHO-OFF	+		+	OPEN-DB
\$	7	+		+	DUMP	+		+	DONE	+		+	EDIT-DB
\$	8	+		+	SET-TITL	+		+		+		+	CLOSE-DB
\$	9	+		+	GET-TITL	+		+		+		+	BEGIN
\$	10	+		+	DONE	+		+		+		+	CONTINUE
\$	11	+		+		+		+		+		+	SYSTEM
\$	12	+		+		+		+		+		+	QUIT-DEX
\$		+	-----	+	-----	+	-----	+	-----	+	-----	+	-----

Figure 1-3. DEX Menus (as printed on terminal)

The user's module is activated when he types,
from menu "DEX.MAIN",

.begin modulename

He then enters the environment of the module, but he can return to the DEX level by using the symbol "\$" and then an item from "DEX.MAIN". Similarly, he can transfer temporarily from the DEX level to the local computer operating system level by the option "system". To get back into DEX he uses the command

.return

and then to get back to the module he uses the menu option "continue".

When a module execution is complete, the user returns to the DEX level and issues the command

.quit

to return permanently to the operating system.

1.2.2.2 Extended DEX Library. The extended DEX library is not a level of operation like DEX on the module, but rather a collection of 45 subroutines and functions which the module author can adopt in constructing his module. The bulk of this investigator's work involved the development of this library, and it will be discussed further in Section 1.3 and Chapters 3-7.

1.2.2.3 Module. A module is a complete set of subprograms written by the module author and executed by the user to perform a specific task. A module may consist of only one program which does the actual calculations, or it may have many additional subprograms employing menus to take advantage of the flexibility offered by DEX and the extended DEX library. Chapter 2 describes in detail an actual module used during the testing of the extended DEX library subprograms, and Chapter 8 describes the Machinery Weight Estimating Module written to demonstrate the use of the cruiser-destroyer databases.

1.3 The Extended DEX Library

In order to convey information from a storage location to the program doing the calculations and from there to another storage location or display, five capabilities are required by the user. These five, listed here, are provided by the extended DEX library:

- (i) Setting and reviewing the module environment for type of dialogue and sources and destinations of information.
- (ii) Reading information.
- (iii) Editing information.
- (iv) Writing information.
- (v) Converting the user-preferred input/output units to the module author-preferred units and back again.

The five categories will be briefly outlined here and described in detail in Chapters 3-7.

1.3.1 Environment. Four subroutines provide or display the module environment defined by the user.

They are:

DIALOG: enables user to specify terse or verbose dialogue

SOURCE: enables user to specify source of information, either a DEX-created database, the terminal, a sequential file or the module's default data.

DESTIN: enables user to specify the destination of information, either a DEX-created database, the terminal or a sequential file

MDMODE: displays the status of the module environment, including type of dialogue, reading source, writing destination, and reference numbers for files to be read from or written to.

1.3.2 Readers. Eight logical functions allow the user to read information from the designated source.

They are:

ISCLDR: invokes ISCPMP and ISREAD

ISCPMP: prepares a prompting message for reading an integer from the terminal

ISREAD: reads an integer value from the source.

RSCLDR: invokes RVAPMP and RSREAD

RVAPMP: prepares a prompting message for reading a real scalar or a real array from the terminal.

RSREAD: reads a real scalar value from the source
RALDR: invokes RVAPMP and RALRED
RALRED: reads a single one-dimensional array from
the source.

1.3.3 Editors. The editing routines are still undergoing development. Eventually they will enable the user to perform various editing functions on the working variables of the module. Two preliminary routines were completed during this investigation:

ISCEDT: enables the user to change the value of an integer scalar variable from the terminal.
RSCEDT: enables the user to change the value of a real scalar variable from the terminal.

1.3.4 Writers. Eight logical functions allow the user to write information to the designated destination. They are:

ISCDMP: calls ISDSCR and ISRITE
ISDSCR: prepares a description message for writing an integer to the terminal
ISRITE: writes an integer scalar to the destination
RSCDMP: calls RVDSCR and RSRITE
RVDSCR: prepares a description message for writing a real scalar or a one-dimensional real array to the terminal

RARDMP: calls RVDSCR and RARITE

RARITE: writes one-dimensional real arrays to the destination

1.3.5 Units. The units subprograms are divided into three categories. The first category contains five subroutines which read, edit or write the input/output units that the user wishes to employ. There is one for each of the five basic types of units adopted by the system: plane angle, force, length, temperature and time. The names of these subroutines are:

AUNIT

FUNIT

LUNIT

TPUNIT

TUNIT

The second category contains five logical functions, one for each of the five basic unit types, which determine the conversion factors for converting to i/o units to the program standard units and vice versa. Additionally, they prepare unit names and abbreviations of unit names which are used in database comments and prompting and description messages. The names of these functions are:

UNITAF

UNITFF

UNITLF

UNITMP

UNITTF

The last category contains twelve logical functions which perform the same function as those in the second category, but for derived units formed by combining basic units. They are:

UAACC: angular acceleration

UACCEL: linear acceleration

UAREA: area

UFREQ: frequency

UKVISC: kinematic viscosity

UMASS: mass

UMPOWER: mechanical power

UPRESS: pressure

UPSPEC: power spectrum

URHO: density

USPEED: speed

UVOL: volume

1.4 Dex Databases

1.4.1 Philosophy. The DEX philosophy includes as a fundamental feature a new and more capable approach to database manipulation. This revolves around the concept of identifying a variable within a database by its name and not by its location. For example, if a user needs the value of an entry in the database signifying the ship's draft, he retrieves that value at the address "DRAFT", or whatever name it has been assigned by the database creator, and not by specifying that the value is the fourth or twentieth entry in the database.

In order to use the capabilities of DEX a database must be constructed via either the commands of menu "DBEDCMDs" or the database management routines in reference [4]. These entail a specific format for the entry of the variable, but there is no sequential order required for arranging the different variables in the database.

1.4.2 Format of Database Entries. In the present version of DEX a database can contain up to 200 variables. Three types of variables are allowed: integer scalars, real scalar, and one-dimensional real arrays. A real array can contain up to 200 elements.

A variable entry in a database consists of four parts. First is the database name, which is formed by

up to eight alphanumeric characters (including blanks), e.g., "LBP", "WEIGHT17", "TYPSONAR". Second is the type of variable - integer, real scalar or real array - and third is the value of the variable itself. The final part is the database comment statement, a string of words up to 64 characters long which describes the variable. If the variable is either a real scalar or real array and has units, the comment statement contains a twelve-character (including blanks) version of the units enclosed in parentheses. Appendix B contains edited listings of a warship general database which illustrates database entries.

Accompanying each database will be a database dictionary which lists for each variable its database name, type, array size, if applicable, and official database comment, including units, if applicable. Future versions of DEX will separate the units from the comment as a distinct fifth part of a variable entry. Further, development has started to create positional databases which will allow database elements to be related to each other, e.g., a database containing a ship's compartments where two compartments are adjacent.

This chapter has attempted to give a brief introduction to the concept and organization of DEX. For the

reader who is confused at this point by the large number of new ideas, terms and subprograms mentioned, Chapter 2 has been included to provide an example of a simple module which employs many of the concepts and subroutines described. It should give the reader a practical awareness of how this all ties together. This will prove helpful in reading the next five chapters which discuss the design of the extended DEX library subprograms. Chapter 8 discusses an application of DEX to computer-aided ship design. Finally, Chapter 9 offers recommendations for future work.

CHAPTER 2

THE CUBE MODULE SAMPLE PROGRAM

2.1 General Description

2.1.1 Function of the Module. The Cube Module was developed to test the subprograms of the extended DEX library written during this investigation. The module calculated the volume and weight of a block of salt water given the length, width, and height (note that "cube" is a slight misnomer). While that function itself was elementary, the combination of single, scalar values for length and width and an array for height (and also, therefore, for volume and weight) permitted the testing of the reading, editing and writing routines for real scalars and the reading and writing routines for real arrays. The subprogram for specifying input and output units employed the routines for integer scalars. The subroutines for determining the conversion factors for length, force and volume were also exercised. Finally, as a matter of course, the environment setting routines were also tested.

Appendix A contains a listing of the module.

2.1.2 Module Subprograms. Although no single, correct sequence of operating the module subprograms existed, there was a logical path one would follow to

execute the program when not attempting to test every available option of the module. This path is a convenient order in which to list the module subprograms and those extended DEX library subprograms involving menus:

MAINPG	(C-M)
DIALOG	(DL-M)
SOURCE	(DL-M)
DESTIN	(DL-M)
MDMODE	(DL)
MODIO	(C-M)
INPUT	(C-M)
MXUNIT	(C-M)
LUNIT	(DL-M)
FUNIT	(DL-M)
DIMENS	(C-M)
COMPUT	(C)
OUTPUT	(C-M)
VANDWT	(C-M)
BLOCK DATA	(C)

The "C" indicates a Cube Module subprogram, the "DL" indicates an extended DEX library subprogram and the "M" indicates that the subprogram included a menu. Figure 2-1 illustrates the menus encountered in this model.

2.1.3 Typical Operation. A description of a typical execution of the Cube Module should prove helpful. To assist the reader, Figure 2-2 shows the various access and return routes of the subprograms. Once the user entered the DEX level environment he activated the Cube Module via the "begin" option of menu "DEX.MAIN", that is

. begin cube

MENU MOD.MAIN	MENU MOD.ALTR	MENU SOURCE	MENU DESTINAT	MENU MOD.IO	MENU INPUT
DIALOGUE	TERSE	DATABASE	DATABASE	ALL	ALL
INMODE	VERBOSE	TERMINAL	TERMINAL	INPUT	UNITS
OUTMODE		SCREEN	SCREEN	OUTPUT	DIMENSIO
MOD.MODE		FILE	FILE	DONE	DONE
READ		DEFAULT			
EDIT					
COMPUT					
WRITE					
QUIT					

MENU UNIT	MENU FOR.UNIT	MENU LEN.UNIT	MENU DIMENSIO	MENU OUTPUT	MENU RESULTS
ALL	POUNDAL	INCH	ALL	ALL	VOLUME
LENGTH	FPOUND	FOOT	LENGTH	UNITS	WEIGHT
TIME	SHORTTON	STATMILE	WIDTH	RESULTS	DONE
FORCE	LONG TON	NAUTMILE	HEIGHT	DONE	
PLANEANG	DYNE	MILLIMET	DONE		
TEMP	NEWTON	CENTIMET			
	KILOPOND	METER			
		KILOMET			

Figure 2-1. Menus Encountered in Executing the Cube Module

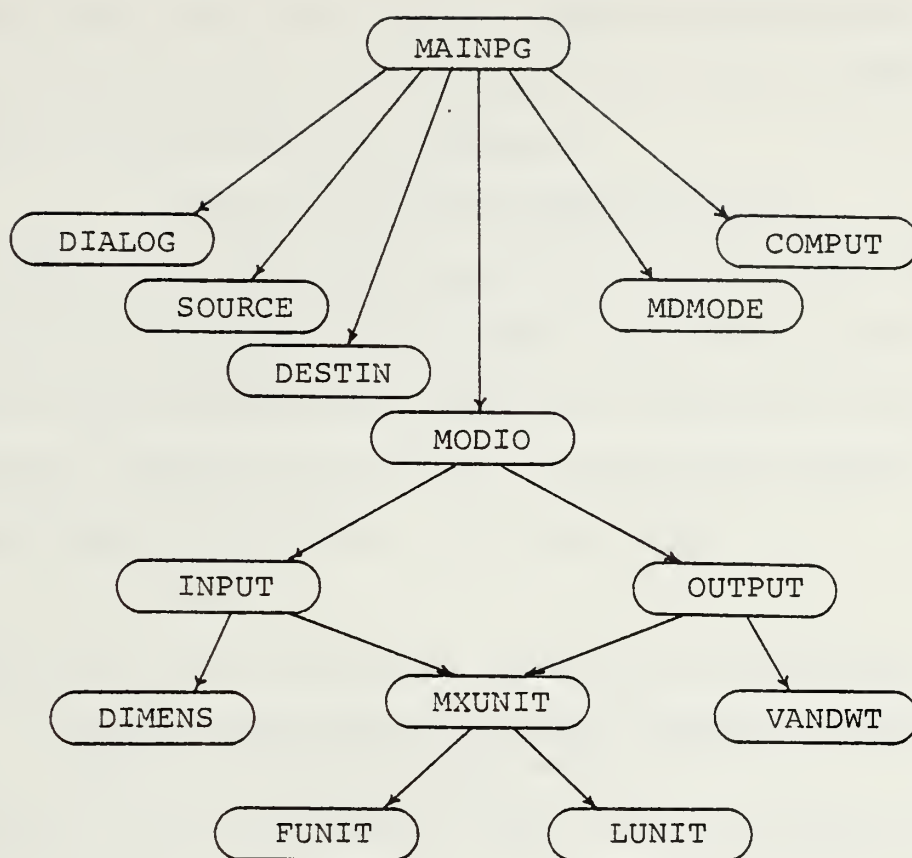


Figure 2-2. Access routes of the various subprograms encountered in executing the Cube Module. Return routes are back along the arrows.

where "cube" was the assigned module identification name. This command placed him in module subroutine MAINPG where he encountered the message:

```
*ENTER AN ITEM FROM MENU-MOD.MAIN
```

The user typed "mod.mode" and was presented with the status of the module environment. The initialized values for the dialogue, source and destination were verbose, terminal and terminal respectively and he found these satisfactory. He then typed

```
.read
```

which sent him to subroutine MODIO and the message

```
*SELECT WHICH MODULE VARIABLE SEGMENT TO READ
*ENTER AN ITEM FROM MENU-MOD.IO
```

From this menu he selected item "input" by typing it:

```
.input
```

Now in subroutine INPUT he received the instruction

```
*SELECT WHICH INPUT VARIABLE SEGMENT TO READ
*ENTER AN ITEM FROM MENU-INPUT
```

At this point, to save time, he made two sequential selections. From menu "INPUT" he selected "units" and anticipating menu "UNIT" he chose "length". He accomplished this by typing

```
.units length
```

DEX recognized the space between the two words as a delimiter between two commands. It acted on the first by invoking subroutine MXUNIT. It then located item

"length" on that subroutine's menu and called subroutine LUNIT which issued the command

```
*ENTER LENGTH UNIT TO BE USED DURING INPUT OUTPUT
*ENTER AN ITEM FROM MENU-LEN.UNIT
```

The user specified "foot". LUNIT then passed control back to subroutine MXUNIT which printed

```
*SELECT WHICH UNIT TO READ
*ENTER AN ITEM FROM MENU-UNIT
```

Again to save time, the user typed in two sequential menu selections:

```
.force fpound
```

This sent him to FUNIT and then back to MXUNIT. This time from menu "UNIT he chose item "done" which returned him to subroutine INPUT. Figure 2-3 illustrates this sequence.

At this point the user had defined the length and weight units he intended to use for input and output. In response to INPUT's request for a menu selection he typed

```
.dimensio
```

This invoked subroutine DIMENS and caused the following to be printed.

```
*SELECT THE DESIRED DIMENSION TO READ
*ENTER AN ITEM FROM MENU-DIMENSIO
```

The user intended to read in all three dimensions (he could have used any or all of the initialized values built into the module in BLOCK DATA) so he typed item


```

*ENTER AN ITEM FROM MENU - MOD.MAIN
.mod.mode
*MODULE DIALOGUE           : VERBOSE
*MODULE INPUT SOURCE       : THE TERMINAL (ALPHANUMERIC)
*MODULE OUTPUT SOURCE      : THE TERMINAL (ALPHANUMERIC)
*REFERENCE NUMBER FORM MODULE
*FORTRAN READ FROM A FILE  : 3
*FORTRAN WRITE TO A FILE   : 4
*ENTER AN ITEM FROM MENU - MOD.MAIN
.read
*SELECT WHICH MODULE VARIABLE SEGMENT TO READ.
*ENTER AN ITEM FROM MENU - MOD.IO
.input
*SELECT WHICH INPUT VARIABLE SEGMENT TO READ.
*ENTER AN ITEM FROM MENU - INPUT
.units length
*ENTER AN ITEM FROM MENU - LEN.UNIT
.foot
*SELECT WHICH UNIT TO READ.
*ENTER AN ITEM FROM MENU - UNIT
.force fpound
*SELECT WHICH UNIT TO READ.
*ENTER AN ITEM FROM MENU - UNIT
.done
*SELECT WHICH INPUT VARIABLE SEGMENT TO READ.
*ENTER AN ITEM FROM MENU - INPUT
.dimensio
*SELECT THE DESIRED DIMENSION TO READ.
*ENTER AN ITEM FROM MENU - DIMENSIO
.all
*ENTER LENGTH OF CUBE (FOOT      )
*ENTER UP TO 1 REAL NUMBERS
.1.0
*ENTER WIDTH OF CUBE (FOOT      )
*ENTER UP TO 1 REAL NUMBERS
.1.0
*ENTER HEIGHT OF CUBE (FOOT     )
*ENTER UP TO 4 REAL NUMBERS
.1. 2. 3. 4.
*SELECT WHICH INPUT VARIABLE SEGMENT TO READ.
*ENTER AN ITEM FROM MENU - INPUT

```

Figure 2-3. Sample Cube Module Input Sequence

"all". The computer responded with:

```
*ENTER LENGTH OF CUBE (FOOT      )  
*ENTER UP TO 1 REAL NUMBERS
```

The user typed in "1.0" and the computer proceeded to the next instruction

```
*ENTER WIDTH OF CUBE (FOOT      )  
*ENTER UP TO 1 REAL NUMBERS
```

The process was repeated, except that for height the computer requested up to four real numbers (height was defined as an array having up to four elements). When the desired number (say four) of heights were entered control returned to INPUT. The user then typed in

```
.done done
```

to return to subroutine MODIO and from there to subroutine MAINPG.

In response to this subroutine's instruction the user typed

```
.compute
```

This invoked the COMPUT subroutine to actually calculate the volumes and weights. Control was then returned to MAINPG as evidenced by the instruction

```
*ENTER AN ITEM FROM MENU-MOD.MAIN
```

By now comfortable with the operation of the module, the user decided to display his results on the terminal. He decided to retain "foot" and "poundforce" as the units, although he could have selected any desired

units by accessing MXUNIT again. The volumes and weights returned by COMPUT were in metric units, that being the system in which COMPUT was written. The conversions took place at the input/output locations. More on this later.

Now observe closely. The user issued the following commands:

```
.write output results all
```

These were selections from menus "MOD.MAN", "MOD.IO", "OUTPUT" and "RESULTS" respectively. The computer printed on the terminal the four volumes and four weights. Figure 2-4 is a copy of that printing.

Satisfied with his answers, the user responded to the current instruction from subroutine OUTPUT with

```
.done done
```

to get back to MOD.MAIN. Choice "quit" at this point caused him to leave the module level and return to the DEX level, facing menu "DEX.MAIN". The "quit" selection allowed him to exit DEX and reenter the operating system level in order to log off.

The rest of this chapter will describe the subprograms of the Cube Module to assist the reader in understanding how to write a module program.


```

*ENTER AN ITEM FROM MENU - MOD.MAIN
wp output results all
*VOLUME OF CUBE (FOOT **3 )
*   INDEX      VALUE
*       1      0.999999E+00
*       2      0.200000E+01
*       3      0.299999E+01
*       4      0.399999E+01
*WEIGHT OF CUBE (POUND(FORCE))
*   INDEX      VALUE
*       1      0.638765E+02
*       2      0.127753E+03
*       3      0.191629E+03
*       4      0.255506E+03

```

Figure 2-4. Sample Cube Results
(as printed on terminal)

2.2 Frequently Used Subroutines

2.2.1 Block Data. We will commence the discussion of the module with the subroutines which may be used with only slight changes in form and/or content in several modules. The user may wish to refer frequently to the listings of these subroutines in Appendix A.

First is a special subprogram, BLOCK DATA, used as standard practice in Fortran to initialize all the labeled common blocks used throughout the module. With respect to input/output variables, a typical labeled

common block appearing in BLOCK DATA and the pertinent subprograms is as follows:

```
COMMON /VINFO/ V(4),VOLNAM,VOLCMT,NDEVF,DEFALV(4)
```

From this we can identify the following information which should appear in BLOCK DATA:

- (i) the variable (dimensioned for its maximum size if an array)
- (ii) the variable database name
- (iii) the variable database comment
- (iv) the number of default values (if it is an array)
- (v) the default value (or dimensioned default array)

These values would all be initialized in the BLOCK DATA. Locating the initialization and dimensioning of all input/output variables in one subprogram facilitates checking and is a highly recommended practice.

The variables are grouped under the subroutines in which they first appeared. MAINPG included four common blocks - REFNOS, INOUTF, DIALFG and MDNCPW. All of these are used in several other subprograms. REFNOS included the variables RNRFIL and RNWFIL which represented respectively the file reference numbers for reading from a sequential file using the Fortran READ command and writing to a sequential file via the Fortran WRITE

command. INOUTF included the variables IMODE and OMODE. The first denoted the source of reading information (1 = database, etc.) and the second denoted the destination for writing information (1 = database, etc.). DIALGF contained the logical variable MTERSE to denote the type of module dialogue (terse or verbose). Finally labeled common MDNCDW contained the integer variable NCPW which was the number of characters per word assumed by the DEX routines. This value was site dependent. For example on IBM computers it was 4. For this reason it was flagged in BLOCK DATA as site-dependent.

The other subroutines which represented the first occurrences of labeled common blocks were LUNIT, TUNIT, FUNIT, AUNIT, TPUNIT, DIMENS and VANDWT. Let us examine DIMENS. It contained nine labeled common blocks. The first four were mentioned above in MAINPG. LUNITS will be described in Chapter 7. The remaining four were listed under subroutine DIMENS in BLOCK DATA. LINFO, WINFO and HINFO contained the variable, database name, comment, default values and number of default values, where applicable, for the length, width and height dimensions respectively. H and DEFALH were dimensioned because they were arrays. The type declarations and dimensions of all the variables were followed by the

data declarations of their values. The remaining common block, DIMFRM, represented the formats to be used when reading from or writing to sequential files. One format was for real scalars and the other for arrays.

2.2.2 MAINPG. Subroutine MAINPG, via menu "MOD.MAIN", allowed the user to select the environment of the module and the desired paths to follow to operate it. The capabilities it gave the user were:

(1) To set the style of module dialogue. This choice invoked subroutine DIALOG.

(2) To select the source of module input. This choice invoked subroutine SOURCE. Remember that "input" here means any information to be read, even output variables.

(3) To select the destination of module output. This choice invoked subroutine DESTIN.

(4) To list the module flags. This invoked subroutine MDMODE which printed the status of the dialogue, the source, the destination, and the file reference numbers.

(5) To access the module reading routines. This choice set the variable IOFLAG=1 and then called MODIO.

(6) To access the module editing routines. This choice set IOFLAG=2 and then called MODIO.

(7) To access the module computing routine by calling COMPUT to execute the calculations.

(8) To access the module writing routines. This choice set IOFLAG=3 and then called MODIO.

(9) To return control to the DEX level and menu "DEX.MAIN". It did this by invoking the DEX routine ENDIT.

The menu name and the items were declared in MAINPG with data statements. DEX routine MENUIN was used to convert the user's selection to an integer flag ITEM for branching to the correct point in MAINPG. MENUIN was the routine that actually printed the prompting instruction to enter a menu item.

The user can make his subroutine MAINPG as simple or extensive as desired within the constraint that the maximum number of menu items (because of MENUIN) is twelve. To show how simple it could be, consider a user who has a program called STRENGTH and for some reason he wanted to start it with the DEX. All he would need in his module besides STRENGTH is the following subroutine:

```
SUBROUTINE MAINPG
CALL STRENGTH
RETURN
END
```

If STRENGTH used menus, DEX routine ENDIT would also have to be called to clear the buffer afterwards.

2.2.3 MODIO. Subroutine MODIO had one parameter in its calling sequence - IOFLAG - which indicated if the operation to be performed was reading, editing and writing. MODIO had two labeled commons - DIALGF and MDNCPW - discussed in BLOCK DATA. DIALGF determined whether the verbose or terse menu prompting message would be issued. NCPW was needed for calling DEX routines STRPAK and LMOVEC.

MODIO presented the user with the menu selections shown in Figure 2-1. The "all" option allowed the user to read, edit or write all the input and output variables. The "all" option was implemented by a logical variable "ALLFLG" which was set to .TRUE if that menu item was chosen. This variable was subsequently passed on through the module. If it was returned .FALSE. to MODIO, it would cause the execution of the last command to be halted and the prompting message for menu "MOD.IO" to be issued so that the user could take corrective action.

2.2.4 MXUNIT and the "ALL" Logic. A subprogram similar to MXUNIT will be needed in any module which allows the user to specify i/o units different from those in which the computing routines were written. MXUNIT permitted the user to read, edit or write the module units he wished to use for input and output. For example, if the user selected "force" from menu "UNIT", subroutine FUNIT

would have been called to present the menu choices for force units. FUNIT and the others in that series will be described in Chapter 7, but some detail is required here because of all the calling parameters involved.

The calling sequence for FUNIT, typical for all five in that series, was as follows:

```
CALL FUNIT(UIFUN,LOCALL,IOFLAG,IOMODE,MTERSE,  
           NCPW,DBFUNN,DBFUNC,PMPREP,PMES,  
           RNFILE,FUNFRM,DEFFUN)
```

The first parameter was an output variable, LOCALL was both input and output, and the rest were all input variables provided by MXUNIT.

UIOFUN would have been assigned a value between 1 and 7 depending on what force unit the user selected.

LOCALL carried the information concerning the "ALL" option. One of the calling parameters for MXUNIT, CALALL, passed on the "ALL" option from subroutine INPUT. If it was .TRUE. then LOCALL would have immediately been assigned .TRUE. also, and the prompting for menu "UNIT" would have been bypassed. Each of the i/o unit-defining routines, such as FUNIT, would have been called and the user asked to specify the desired choice of the particular type of unit. Even if CALALL was .FALSE., the user could have selected "all" from menu "UNIT" with similar results.

If any of the i/o unit-defining subroutines was unsuccessful, LOCALL would have been returned .FALSE.. The program would have checked the value of CALALL. If it too was .FALSE., the menu "UNIT" would have been presented for another selection. However, if CALALL was .TRUE., it would have meant that there had been a change in the value of LOCALL (i.e., a failure in the called subprogram). MXUNIT would have set CALALL to .FALSE. and returned control through subroutine INPUT back to MODIO. This was because INPUT incorporated the same "ALL" logic as MXUNIT.

Note from the listing of MXUNIT that LOCALL is initialized as .FALSE.. If CALALL was .FALSE. and the user did not choose "all" from menu "UNIT", he would have been asked to select from the menu each time after a unit was specified, until he typed "done". This "ALL" logic was used extensively throughout the module as a means of expediting the reading, editing or writing of many variables.

Returning to the calling sequence for FUNIT, we have already mentioned IOFLAG, indicating the operation to be performed. PMPREP was set locally by a data declaration. It indicated to the subprograms called whether or not a prompting message for the unit to be selected was to be prepared by the program. If .TRUE., PMES would later be the storage location for the prompting message. If

PMPREP was .FALSE., PMES would already have to have the prompting message in it. Since PMES first appeared here in MXUNIT it was dimensioned here for the maximum allowable size of 16 "words". This number came from the limitation on PMES to be at most 64 characters long, divided into 4-character "words".

All the other variables were obtained by MXUNIT from other subprograms, including BLOCK DATA, by labeled common blocks. An inspection of the listing of MXUNIT reveals the large number of commons required because of five types of units.

INOUTF passed the indicator of the source of information to be read (IMODE) and the destination of information to be written (OMODE). Depending on IOFLAG, IOMODE was set equal to one of these two. This told FUNIT from where UIOFUN was to be read or to where UIOFUN was to be written.

REFNOS passed the values of the file reference numbers for Fortran READ (RNRFIL) and Fortran WRITE (RNWFIL). Depending on IOFLAG, RNFILE was set equal to one or the other. These values are assigned to files during the execution of extended DEX library routines SOURCE and DESTIN by DEX routine SETDEV if the user had designated files as the source or destination.

The other labeled commons will be described in Chapter 7.

2.3 The Input Series

2.3.1 INPUT. Subroutine INPUT provided the user with a menu by which he could access subroutines MXUNIT and DIMENS and return control to subroutine MODIO. It contained the labeled common blocks DIALGF and MDNCPW for use in calling STRPAK and LMOVEC for character manipulation. INPUT had two variables in its calling sequence - CALALL and IOFLAG - as did MXUNIT described above. The logic for the use of CALALL and LOCALL was identical to that described in 2.2.4. Briefly, if LOCALL was set to .TRUE. when INPUT was invoked because CALALL=.TRUE., and it was returned by MXUNIT or DIMENS as .FALSE., CALALL would have been set equal to .FALSE. and control passed back to MODIO. If CALALL was .FALSE., LOCALL would have been .FALSE. unless the user selected "all" from menu "INPUT".

Besides LOCALL, INPUT passed IOFLAG to MXUNIT and DIMENS to indicate reading, editing or writing.

2.3.2 DIMENS. Subroutine DIMENS allowed the user to read, edit, or write the value of the block dimensions. It had two calling parameters, ALLFLG and IOFLAG. The former carried the "ALL" option information and behaved like CALALL. The "ALL" option was passed on to the called logical functions by the usual variable LOCALL.

DIMENS, like MXUNIT, also contained quite a few labeled common blocks. Five have already been seen in

MXUNIT: DIALGF, INOUTF, MDNCPW, REFNOS, and LUNITS. LINFO, WINFO, and HINFO contained the variable, database name, comment and default values for the length, width and height of the block. DIMFRM contained the format information for reading from or writing to files.

Before menu "DIMENSIO" was presented to the user, two actions occurred. The first was the calling of the logical function UNITLF by the statement

```
LOGVAL=UNITLF(CONVLM,NAMLØ2,NAMLØ6,NAML12,ALLFLG,  
              PSTLUN,UIOLUN,NCPW)
```

UNITLF, discussed in Chapter 7, produced the multiplicative conversion factor CONVLM for converting the dimensions in the input length unit to values in the program standard length unit (meter). The second through fourth variables in UNITLF's calling sequence represented various abbreviations of the length unit the user had selected for input/output. UNITLF was able to provide this information because of the values of PSTLUN and UIOLUN. The first indicated program standard length unit, the second user i/o length unit. As a pair they were an index to locating the other information.

The other action which occurred immediately in DIMENS was the setting of LOCALL equal to ALLFLG. If this made LOCALL equal to .TRUE., menu "DIMENSIO" was not presented and DIMENS immediately started operating on all the menu choices.

If the user selected "width" from the menu, the program next referred to IOFLAG. If IOFLAG=1 (reading), the program branched to the statement

```
LOGVAL=RSCLDR(W,LOCALL,MTERSE,IMODE,NCPW,WIDNAM,  
              CONVLM,CONVLA,NAML12,.FALSE.,.TRUE.,  
              PMES,WMORGN,RNRFIL,LWFRM,DEFALW)
```

Logical function RSCLDR is the first of three functions for reading real scalars. The value read in for width, in program standard units, was stored in W. IMODE indicated the source of the reading. WIDNAM was the 8-character database name for width. CONVLM and CONVLA were respectively the multiplicative and additive conversion factors for changing the read in width to the value in program standard units via the expression

$$W=W * CONVLM + CONFLA$$

CONVLA was locally declared 0. NAML12 was the 12-character version of the i/o length unit and was used in prompting message preparation and database comment comparison. The parameter .FALSE. represented the variable VITAL which indicated if the variable W was essential for input continuation. The parameter .TRUE. was for PMPREP, indicating that the reading subprograms would prepare the prompting message PMES, at this point undefined. Since PMES was a local variable not passed to DIMENS by the calling sequence or a labeled common block it was dimensioned "16" in DIMENS. WMORGN contained the description

of the width, including space allotted for inserting the units. RNRFIL and LWPRM were the reference number and format respectively for reaching width from a file. DEFALW was the default value for width if the user wished to set W equal to that.

If IOFLAG=2, the subprogram called the editing routines by the expression

```
LOGVAL=RSCEDT(W,LOCALL,MTERSE,NCPW,WIDNAM,CONVLM,  
              CONVLA,NAML12,.TRUE.,PMES,WMORGN,  
              RNRFIL,LWFRM,DEFLAW)
```

This was very similar to the reading function except that IMODE was not specified. This was because RSCEDT itself defined and employed a variable EDMODE, of value 2, for the terminal, in lieu of IMODE, when it called RSCLDR.

Finally, if IOFLAG=3, the real scalar writing routines were invoked by the statement

```
LOGVAL=RSCDMP(LOCALL,MTERSE,OMODE,NCPW,W,WIDNAM,  
              CONVLM,CONVLA,NAML12,.TRUE.,PMES,  
              WMORGN,RNWFIL,LWRFM)
```

Observe that OMODE was used to indicate destination of the value of W for writing. Also note the use of RNWFIL to indicate the file reference number for writing with Fortran WRITE, using the format supplied by LWFRM.

Because the height variable H represented a four-element array, the logical functions called were different. For reading (IOFLAG=1) DIMENS branched to the statement


```
LOGVAL=RALLDR(H,LOCALL,NGOT,MTERSE,IMODE,NCPW,HEINAM,  
MXTOGT,CONVLM,CONVLA,NAML12,.FALSE.,  
.TRUE.,PMES,HMORGN,RNRFIL,HFRM,NDEFH,  
DEFALH)
```

Several new variables appear here. MXTOGT represented the maximum number of elements to extract from the source when the source was a database or the terminal. NGOT represented the actual number of elements read from the source. Both MXTOGT and NGOT were initialized as 4 in DIMENS. VITAL was defined by the .FALSE. and PMPREP by the .TRUE.. HFRM contained the format for reading the array from a file. NDEFH and DEFALH were respectively the number of default values and a four-element array containing the default values for height.

Because the array editing routines had not yet been written, provision for calling a dummy routine, RAREDT, was included in DIMENS.

```
For writing (IOFLAG=3), the program branched to  
LOGVAL=RARDMP(LOCALL,MTERSE,OMODE,NCPW,H,HEINAM,  
NFROM,NGOT,CONVLM,CONVLA,NAML12,  
.TRUE.,PMES,HMORGN,RNWFIL,HFRM)
```

NGOT, mentioned above, and NFROM, were initialized as 4 and 1 respectively in DIMENS. NGOT could have a value different from 4 only if less than four elements were read in when RALLDR was called.

2.4 The Output Series Subprograms

The output series consisted of two subprograms - OUTPUT and VANDWT - for working with the volumes and weights calculated by COMPUT. These had direct parallels to INPUT and DIMENS and need not be discussed in detail. OUTPUT offered the user the capability of invoking MXUNIT, via the menu item "unit", in case the user wished to write the volume and weight in different units from those he had used for reading in the block dimensions.

2.5 General Programming Comments

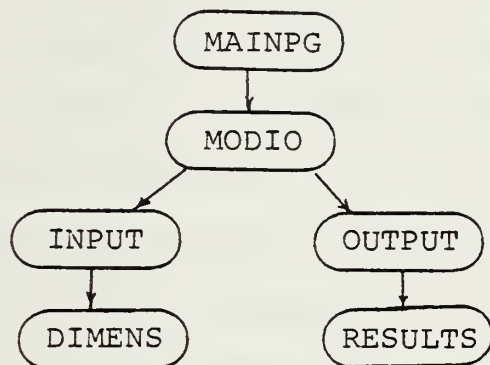
This chapter will conclude with some comments about writing modules. It is hoped that the reader has some understanding of the calling parameters used by module subprograms when invoking extended DEX library routines. In some cases large numbers of variables appear in the calling sequences. This is due to the fact, as will be pointed out in the next chapter, that the library routines use no common blocks.

One suggestion already emphasized is the use of a BLOCK DATA subprogram to initialize all input/output variables and their associated variables. This adds some extra work by increasing the number of common block variables, such as the database names, comments and default values. However, the improved efficiency for checking values and dimensions is considered to outweigh this disadvantage.

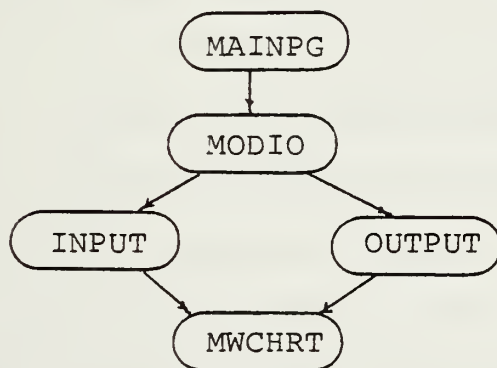
When writing a module, to determine the menus required it is easiest to work backwards in the opposite direction to the order of use. Identify the input and output variables and place them in menus. Then establish a supervisory input menu and a supervisory output menu. For example, one may contain the group name for the input variables, such as "dimensions" in Cube, plus a units item to allow the user the choice of i/o units. At this point the module author may be almost done with the module design phase, because he can frequently incorporate the standard routines MAINPG and MODIO to complete his set. MXUNIT is also offered as a very adaptable, comprehensive routine for handling units. In a situation like the Machinery Weight Estimating Module in Chapter 8, one menu suffices for both input and output variables. Figure 2-5 illustrates the difference in flow between the main subroutine and the i/o variables for the two modules. Yet both use identical MAINPG and MODIO subprograms and only slightly different versions of MXUNIT.

When establishing menus a few rules must be kept in mind. The maximum number of menu items is twelve. When constructing the eight character menu item names, no blanks are allowed between characters but are acceptable after all the characters. If each menu item begins with a different character, only that one character has to be

entered by the user to enable DEX routine MENUIN to identify the selection.



Cube Module



Machinery Weight Estimating Module

Figure 2-5. Comparison of Module Input/Output Flow

CHAPTER 3

THE EXTENDED DEX LIBRARY ENVIRONMENT SETTING ROUTINES

3.1 Introduction

Upon entering the module level of DEX operation, the user needs to establish or verify the environment which suits his requirements. Four extended DEX library subroutines give him the capability to do this. They are:

- (1) DIALOG, which sets the type of module dialogue
- (2) SOURCE, which defines the location of information to be read
- (3) DESTIN, which defines the location to which information is to be written
- (4) MDMODE, which displays on the terminal the current status of the type of dialogue, the source, the destination and the file reference numbers for Fortran READ and WRITE to sequential files.

Each of these will be discussed in this chapter. Logical function CHKRNG, revised during this investigation, is also discussed here, although it will eventually become a DEX routine and not an extended DEX library function.

3.2 Subroutine DIALOG

3.2.1. Menu and Calling Parameter. Subroutine DIALOG would normally be called by a subroutine similar to the

subroutine MAINPG of the Cube Module. In that specific case it was done by selecting item "dialogue" from menu "MOD.MAIN". DIALOG has its own menu, and this is illustrated in Figure 3-1. Note the absence of an item "done",

```
$      + ----- +
$      +   MENU   +
$      + MOD.ALTR +
$      + ----- +
$  1  +  TERSE   +
$      + ----- +
$  2  +  VERBOSE +
$      + ----- +
```

Figure 3-1. Menu "MOD.ALTR"
(for Subroutine DIALOG)

which appeared in most of the Cube Module menus. This is typical for the subroutines of this chapter. Since the user only makes one choice from these menus, the subprograms automatically return control to the calling program (e.g. MAINPG) without further action by the user.

It should also be called to the reader's attention that the DEX library subroutines employ no labeled common blocks. Rather, all variable values are transmitted by means of the calling sequences. This was done to minimize the possibility of inadvertently passing improper values among the many subprograms which share the same commons. The calling sequence presents a readily-checked format for tracing errors.

The only variable in the calling sequence for DIALOG is the logical variable MTERSE. If MTERSE=.TRUE., the dialogue type is terse; it is is .FALSE. the dialogue is verbose. These two types of dialogues are offered to satisfy the needs of the individual user. For the new user, the verbose dialogue provides longer messages from the module or DEX to facilitate learning how to use them. The terse dialogue allows more rapid operations by the experienced user.

Initialized in BLOCK DATA, MTERSE can have its value changed by the correct selection from menu "MOD.ALTR". This is accomplished by subroutine MENUIN. MENUIN is a

DEX integer function (see reference [5]) which converts a menu selection into an integer value. In DIALOG, the following statements are involved in this process:*

```
DATA LMS/8/
DATA MENUNM/4HMOD.,4HALTR/
DATA NITEMS/2/
DATA ITEMS/4HTERS,4HE    ,
1      /4HVERB,4HOSE /
CALL STRPAK (MESS,LMS,4H<    ,29HSELECT TYPE OF MOD
1ULE DIALOGUE<)
ITEM=MENUNM(NITEMS,ITEMS,MESS)
```

MENUNM prints both the prompting message MESS shown above and the message

*ENTER AN ITEM FROM MENU - MOD.ALTR

MENUNM sets ITEM equal to 1 if the user selects TERSE and 2 if he selects VERBOSE. ITEM is then used to branch to statements which make MTERSE either .TRUE. or .FALSE. as appropriate. Both branches return control to the calling program.

3.3 Subroutine SOURCE

3.3.1 Menu and Calling Sequence. Subroutine SOURCE, normally accessed from a subprogram like MAINPG, allows the user to select from the menu shown in Figure 3-2 the source of information to be read. Subroutine MENUNM assigns a value between 1 and 5 to IMODE depending on the user's choice.

* STRPAK is a DEX routine which inserts character strings into an integer array in the DEX format of four characters per word. See ref [5].

The calling sequence for SOURCE is as follows:

```
SUBROUTINE SOURCE(IMODE,DBFNME,IFILNM,RNRFIL,  
                  MTERSE,NCPW)
```

DBFNME is the database filename when the source is a database. IFILNM is the name of the sequential file if the user intends to read from a file. RNRFIL is an input variable defined in BLOCK DATA which represents the device number to be assigned to file IFILNM. The information in the file will be read in by one of the reading routines using the statement of the form

```
READ (RNRFIL, FORMAT) X
```

where X represents the variable being read. MTERSE and NCPW are required here for the preparation of the menu prompting message and other error messages supplied by subroutine source.

```
$      + ----- +  
$      +  MENU   +  
$      + SOURCE  +  
$      + ----- +  
$  1 + DATABASE +  
$      + ----- +  
$  2 + TERMINAL +  
$      + ----- +  
$  3 + SCREEN   +  
$      + ----- +  
$  4 + FILE     +  
$      + ----- +  
$  5 + DEFAULT  +  
$      + ----- +
```

Figure 3-2. Menu "SOURCE"

3.3.2. Operation of SOURCE. SOURCE commences execution by calling MENUIN to prompt the user to make a menu selection. MENUIN assigns an appropriate value to IMODE for branching.

If IMODE=1, the subroutine requires a database via DEX logical function DBOPEN. If there already is an open database, DBOPEN asks the user if he wants to use it, and if so, if he wants to save it ("save" has the usual meaning of writing a copy into memory from the buffers) before using it. If a database is "active", meaning a copy is in the buffers but there is also a saved copy in memory, the user is asked only if he wants to use it. If the user answers "no" to either of these questions, or if there is no active or open database, DBOPEN prompts the user for the name of either a new one to be created or an existing one to be opened. DBFNME is assigned that name. When this is done control is returned to the calling program.

If IMODE=2 (terminal) or 5 (default values) control simply returns to the calling program. If IMODE=3, indicating that the user hoped to employ DEX routines to read X-Y coordinates from a screen plot, he is informed that this mode of module input has not been implemented yet.

IMODE=4 causes the calling of subroutine GETFLNM which asks the user the name of the sequential file to be

read. Then logical function SETDEV assigns RNRFIL to IFILNM before control returns to the calling program.

3.4 Subroutine DESTIN

3.4.1 Menu and Calling Sequence. Subroutine DESTIN is similar to SOURCE except that there are only four possible choices, as seen in the menu illustration in Figure 3-3. The calling sequence contains six parameters:

```
SUBROUTINE DESTIN(OMODE,DBFNME,OFILNM,RNWFIL,  
                  MTERSE,NCPW)
```

OMODE is assigned a value between 1 and 4 by MENUIN. DBFNME is the same as in SOURCE. OFILNM is the name of the sequential file to which output is to be written. RNWFIL is an input variable defined in BLOCK DATA which represents the file reference number to be assigned to OFILNM for Fortran WRITE.

3.4.2 Operation of DESTIN. This subroutine behaves very similarly to SOURCE. If OMODE=1, DBOPEN checks for and opens as necessary a database and assigned DBFNME its name. If OMODE=2, control simply returns to the calling program. If the user selected the screen in order to do plotting, he is informed that this mode of output has not yet been implemented and is asked to make another selection. If OMODE=4, the user is asked the file name and it is assigned to OFILNM. Then logical function SETDEV assigns RNWFIL to the file and control is returned to the calling program.


```

$      + ----- +
$      +   MENU   +
$      + DESTINAT +
$      + ----- +
$  1  + DATABASE +
$      + ----- +
$  2  +  TERMINAL +
$      + ----- +
$  3  +   SCREEN  +
$      + ----- +
$  4  +    FILE   +
$      + ----- +

```

Figure 3-3. Menu "DESTINAT"
(for Subroutine DESTIN)

3.5 Subroutine MDMODE

3.5.1 Function. Subroutine MDMODE informs the user of the current value of the following variables:

- (1) MTERSE, which denotes the type of module dialogue
- (2) IMODE , which denotes the source of information to be read
- (3) OMODE , which denotes the destination of information to be written
- (4) RNRFIL, which denotes the file reference number for module Fortran READ from a sequential file
- (5) RNWFIL, which denotes the file reference number for module Fortran WRITE to a sequential file

After displaying this information, or an error message if a failure occurs, MDMODE returns control to the calling program.

3.5.2 Operation of MDMODE. The calling sequence for MDMODE contains only variables already discussed.

```
SUBROUTINE MDMODE (IMODE,OMODE,RNRFIL,RNWFIL,  
                  MTERSE,NCPW)
```

MDMODE does not display the numerical values of IMODE and OMODE but rather, for the user's convenience, it prints character strings defining the source and destination, e.g. "a dex created database". Similarly, for MTERSE, it prints "terse" and "verbose" vice ".TRUE." or ".FALSE.".

3.6 Logical Function CHKRNG

Logical function CHKRNG determines if an integer number is within the range defined by two other integer numbers. If not, an appropriate error message is issued and CHKRNG is returned .FALSE.. The calling sequence is as follows:

```
LOGICAL FUNCTION CHKRNG (NUMBER,MNEMON,MINNUM,  
                        MAXNUM,NCPW)
```

The routine checks if NUMBER is between the lower number MINNUM and the higher number MAXNUM. MNEMON is an 8-character mnemonic for NUMBER used in the error message. The use of CHKRNG avoids the need for the module author to include a message in his program.

CHAPTER 4

THE EXTENDED DEX LIBRARY READING ROUTINES

4.1 General Description

4.1.1 Function. Information, which includes both input values and previously calculated output values, resides in four locations: DEX-created databases, the user himself, requested files and default data stored in the module. The user actually presents two distinct sources of information to the computer because of the two ways in which he can transmit data at the terminal: the first in the form of alphanumeric characters and the other by the location of a pen-pointer or cross-hairs on a plot on the screen. The latter method has not yet been implemented in DEX at MIT.

The function of the extended DEX library reading routines is to permit the transmission of that information to the module so that it can be written to other locations, such as the terminal for inspection, and/or used as input for the calculations being performed.

4.1.2 Organization. Eight logical functions comprise the reading routines group. They are listed here by the type of variable on which they operate:

<u>Integer Scalar</u>	<u>Real Scalar</u>	<u>Real Array</u>
ISCLDR	RSCLDR	RA1LDR
ISCPMP	RVAPMP	
ISREAD	RSREAD	RA1RED

The real scalar and array categories share RVAPMP.

These routines could be categorized horizontally by their specific jobs, as evidenced by the similarities in their names. The top three are called "loaders". These are the functions that appear in the module sub-programs where it is desired to read variables, and to the module author, for all intents and purposes, they are the reading routines. He does not need to be aware that they actually call the others to do the work.

If the user intends to input from the terminal he needs to be prompted for the correct information. This prompting message can be supplied by the module or it can be prepared by two routines in this group called "prompters". The loaders call the prompters as they are required. Having ensured that a prompting message exists, the loaders then call the third category, the "readers", to actually read in and assign the values to the input. output and working variables.

4.2 Integer Scalar Series

4.2.1 ISCLDR

4.2.1.1 Calling sequence. The calling sequence for logical function ISCLDR includes 17 variables listed here:

```
LOGICAL FUNCTION ISCLDR(IVAR,ALLFLG,  
                        MTERSE,IOMODE,NCPW,DBNAME,VITAL  
                        MENUFL,MENUNM,NITEMS,ITEMS,PMPREP,  
                        PMES,PMORGN,RNFILE,FORMAT,DEFAULT)
```

From the point of view of the function, IVAR is an output variable, ALLFG is both input and output, and the remainder are all input variables.

IVAR is where the value of the integer sought will be stored. VITAL should be discussed next. An essential input variable, that is one required when reading input so that subsequent values can be entered correctly, is indicated when the logical variable VITAL has a value .TRUE.. This is important to the value of ALLFG. ALLFLG indicates the status of the calling program "all" option (active when ALLFLG=.TRUE.). If both VITAL and ALLFLG are .TRUE. and IVAR is not read successfully, or the prompting message is not prepared successfully, ALLFLG will be changed to .FALSE.. ALLFLG's value can also be changed to .FALSE. during a reading evolution if IOMODE=4 (file source) and a premature end-of-file is encountered. Otherwise ALLFLG will retain its input value.

MTERSE indicates whether the module dialogue is terse or verbose. IOMODE denotes the source of input and corresponds to IMODE in the calling program. NCPW is the number of characters per word assumed by the DEX routines. DBNAME is where the database name of the integer is stored. Eight characters (including blanks) must be defined for this variable.

MENUFL is a logical variable which indicates if the integer sought will be a menu selection. If it is .TRUE., the next three variables must be defined. MENUNM is the eight-character name of the menu from which the selection will be made. NITEMS is the number of items in the menu. In the current version of DEX, the maximum number of menu items allowed is 12. ITEMS are the menu choices. Each choice is described by a name of eight characters (including blanks). Therefore, with four characters per word, ITEMS must be dimensioned as twice NITEMS where it first appears. The reader may wish to refer to the AUNIT series routines of Chapter 7 as examples of where menus are used to define integer values.

PMPREP is a logical variable which, when .TRUE., indicates that function ISCPMP will prepare the prompting message for the menu or INTIN. INTIN is a DEX routine which reads integer values entered from the terminal.

If PMPREP=.FALSE., the calling program supplies the prompting message in PMES.

PMES can be up to 64 characters long, and if less than 64 characters it must include the trim character, "<", last to signal the end of the string. Note that if PMPREP=.TRUE., PMES is undefined in the calling sequence of ISCLDR.

PMORGN ("prompting message origin") is a character string, usually the database comment, which identifies the variable sought. Like PMES, it must be 64 characters or less long and must include the trim character at its end if less than 64. When IOMODE=1, PMORGN will be compared with the database comment and differences brought to the user's attention.

RNFILE, corresponding to RNRFIL of the calling program, is the device number for Fortran READ if the integer is to be read from a file. FORMAT is the format for reading from the file. DEFAULT is the default value of the integer sought when IOMODE=5.

4.2.1.2 Characteristics. When the source of the integer is the terminal, ISCLDR invokes routine CHKRNG to verify that NITEMS is between 1 and 12 inclusive when MENUFL=.TRUE.. If PMPREP=.TRUE., ISCLDR invokes ISCPMP to prepare the prompting message. If ISCPMP is returned .FALSE., ISCLDR is also set to .FALSE. and

control returns to the calling program. Otherwise, for terminal input, as well as database, file and default value input, ISCLDR calls logical function ISREAD to do the reading. ISCLDR is set to .TRUE. or .FALSE. depending on the similar value of ISREAD and control returns to the calling program.

ISCLDR becomes .FALSE. if ISCPMP or ISREAD fails. It also becomes .FALSE. if the programmer has bypassed previous checks and set IOMODE=3. The user is informed by ISCLDR that integer scalars cannot be read in from a screen supplying x-y coordinates. The subprogram then checks VITAL and if .TRUE., it advises the user that the variable is essential for program continuation and must be corrected. Then, if ALLFLG=.TRUE., it is changed to .FALSE. and the user is advised that the calling program "all" option is aborted. ISCLDR is set to .FALSE. and control returns to the calling program.

4.2.2 ISCPMP

4.2.2.1 Calling sequence. Logical function ISCPMP is used to prepare a prompting message suitable for identifying the integer being sought when the source is the terminal (IOMODE=2) and PMPREP=.TRUE.. The calling sequence for this function is as follows:

```
LOGICAL FUNCTION ISCPMP(PMES,ALLFLG,MTERSE,NCPW,  
                        DBNAME,VITAL,MENUFL,PMORGN)
```


These parameters have been described in section 4.2.1.1. PMES is the output variable where the prompting message sought is stored. MTERSE is needed here to determine whether a brief or long message is to be prepared. MENUFL is included here but presently it is not used by ISCPMP.

4.2.2.2 Characteristics. ISCPMP creates the prompting message by inserting the word "ENTER", followed by a short or long description of the variable depending on MTERSE, into PMES. When the dialogue is verbose, PMORGN is used as the indentifying string. The program scans it for the location of the trim character to determine its length. If the string is 51 characters or less, the message can be fit on one line. "ENTER" and "PMORGN" are copied into PMES. If, however, PMORGN has more than 64 characters, the prompting message must be two lines long. The word "ENTER" is printed immediately and PMORGN alone is copied onto PMES, to be printed by ISREAD.

When the dialogue is terse, PMES is created from "ENTER" and the database name, DBNAME, with a trim character added at the end.

If a failure occurs in preparing the prompting message, an error message is used. Then, when VITAL=.TRUE., the user is advised that the variable is

necessary for program input continuation and the problem must be corrected. When both VITAL and ALLFLG are .TRUE., the later changes to .FALSE. and the program advises the user that the calling program "all" option is aborted. Finally, ISCPMP is set to .FALSE. and control returns to ISCLDR. If the message preparation is successful, ISCPMP is set to .TRUE. before returning control to ISCLDR.

4.2.3 ISREAD

4.2.3.1 Calling sequence. Logial function ISREAD actually does the reading of the integer from the source defined by IOMODE. The calling sequence for ISREAD is as follows:

```
LOGICAL FUNCTION ISREAD(IVAR,ALLFLG,  
                        MTERSE,IOMODE,NCPW,DBNAME,VITAL  
                        MENUFL,MENUNM,NITEMS,ITEMS,  
                        PMES,RNFILE,FORMAT,DEFAULT)
```

It is almost identical to ISCLDR, the difference being that PMPREP is no longer needed. PMES should be defined here and it must include the trim character if not 64 characters long.

4.2.3.2 Characteristics. ISREAD branches depending on the value of IOMODE. If the source of the integer is a database (IOMODE=1), ISREAD extracts the value using the DEX integer function IGET (see reference [5]). IGET provides error codes which, if other than

success, cause appropriate error feedback messages to be issued.

When IOMODE=2 (terminal input), and MENUFL=.TRUE., routine MENUIN is invoked to obtain IVAR from the menu selection. When a menu is not employed, DEX routine INTIN prompts the user to supply the integer value and reads what is entered at the terminal.

For IOMODE=4 (file source), ISREAD uses a Fortran READ statement, involving RNFIL and FORMAT, to read from the file. The program issues a warning if a premature end-of-file is encountered.

Whenever an error occurs in the reading, or if the user insists on trying IOMODE=3, VITAL and ALLFLG are checked. A warning that the variable is essential for program continuation is issued when VITAL=.TRUE.. ALLFLG will then be set to .FALSE. if not already. ALLFLG's value will also be changed, even if the variable is not essential, if either no open database was found (IOMODE=1) or a premature end-of-file is encountered (IOMODE=4), since further attempts at reading from these sources would be fruitless. In all cases of errors, ISREAD is set to .FALSE. and control is returned to the calling program. When the reading is successful, ISREAD is set to .TRUE..

4.3 Real Scalar Series

4.3.1 Brief description. The function of this group of routines is to permit the user to read real scalar values from any of the designated sources. Three logical functions make up this series: RSCLDR, RVAPMP and RSREAD. RVAPMP prepared prompting messages for reading both real scalars and real arrays from the terminal.

4.3.2 RSCLDR

4.3.2.1 Calling sequence. Logical function RSCLDR is invoked from the module. Its calling sequence is as follows:

```
LOGICAL FUNCTION RSCLDR(RVAR,ALLFLG,  
                        MTERSE,IOMODE,NCPW,  
                        DBNAME,UNITFM,UNITFA,UNITNM,VITAL  
                        PMPREP,PMES,PMORGN,RNFILE,FORMAT,  
                        DEFAULT)
```

Many of these variables are identical to those used in ISCLDR.

RVAR will be assigned the value, in program standard units, of the real number to be read. UNITFM and UNITFA are respectively the multiplicative and additive conversion factors which convert the real scalar read from the user input/output units to the program units. The conversion occurs in RSREAD. UNITNM is a 12-character version (including blanks) of the user i/o units, which is used in preparing the prompting message. If the real

scalar is not dimensional, UNITFM must be equal to 1.0, UNITFA must equal 0.0 and UNITNM is undefined.

PMORGN contains a string of 64 characters or less which identify the variable sought. If it has less than 64 characters it must have the trim character at the end of the string. If the real variable has units, PMORGN should contain the string "(????????????)" into which UNITNM is inserted at the appropriate time. PMES must be dimensioned sufficiently large to accomodate PMORGN plus 6 characters (for "ENTER ") or 64 characters, whichever is less.

4.3.2.2 Characteristics. RSCLDR behaves similarly to ISCLDR. If the source is the terminal and PMPREP=.TRUE., it calls RVAPMP to prepare a prompting message. When the preparation is unsuccessful, RVAPMP is returned .FALSE.. RSCLDR becomes .FALSE. also and control returns to the calling program. If, however, the message preparation is successful, or for IOMODE=1, 4 or 5, RSCLDR calls RSREAD to read the real value. RSCLDR is set to .TRUE. or .FALSE. depending on the success or failure of RSREAD.

When IOMODE=3, RSCLDR informs the user that reading x-y coordinates from a graph on the screen is inappropriate for real scalar input. If VITAL=.TRUE., it issues an

advisory message to the user that the variable is essential for program continuation. ALLFLG is changed to .FALSE. if not already, with a warning that the "all" option is aborted, RSCLDR is set to .FALSE., and control returns to the calling program.

4.3.3 RVAPMP

4.3.3.1 Calling sequence. Both RSCLDR and RALLDR invoke RVAPMP to prepare a prompting message for real scalars and real arrays respectively. The calling sequence is:

```
LOGICAL FUNCTION RVAPMP(PMES,ALLFLG,  
                        MTERSE,NCPW,DBNAME,UNITNM,VITAL,  
                        PMORGN)
```

These parameters are identical to those for ISCPMP except for UNITNM which carries the user i/o unit to be inserted into PMES.

4.3.3.2 Characteristics. When RVAPMP is invoked, PMORGN is scanned for the location of the trim character and for the string "(????????????)" called UNITPT. The presence of UNITPT indicates that the variable is dimensional.

When the dialogue is verbose, the prompting message will be one line long if the trim character is found before the 59th position. The word "ENTER " and PMORGN are copied into PMES, and UNITNM is inserted

into UNITPT if applicable. This is why UNITNM must be 12 characters long. If PMORGN is longer than 58 characters, the prompting message will be two lines long. The string "ENTER" is printed immediately and PMORGN alone is copied into PMES, corrected by UNITNM if necessary. PMES will be the second line of the prompting message.

When the dialogue is terse, PMES is created from the word "ENTER ", followed by DBNAME, UNITPT adjusted by UNITNM, and lastly a trim character.

If the message preparation is successful, RVAPMP is set to .TRUE. and control returns to RSCLDR or RALLDR as applicable. If, however, an error occurs, VITAL is checked. The user is advised that the variable is essential when VITAL=.TRUE.. If ALLFLG also is .TRUE., it is changed to .FALSE. and the user told the "all" option is no longer in effect. In all cases of error, RVAPMP is returned as .FALSE. to the calling program. When successful, RVAPMP is set to .TRUE. before returning control.

4.3.4 RSREAD

4.3.4.1 Calling sequence. Logical function RSREAD reads the real scalar sought from one of the four valid sources of information. It includes 13 parameters in its calling sequence, almost all of which are identical to those in RSCLDR. The sequence is listed here:

LOGICAL FUNCTION RSREAD(RVAR,ALLFLG
MTERSE,IOMODE,NCPW,DBNAME,
UNITFM,UNITFA,
VITAL,PMES,RNFILE,FORMAT,DEFAULT)

Note the absence of UNITNM, PMPREP and PMORGN. These variables have either been used or incorporated into PMES, which is defined at this point.

4.3.4.2 Characteristics. If the source of information is indicated to be the user hoping to input x-y coordinates from the screen, he is informed that this mode of input is inappropriate for real scalar input. The usual checks of VITAL and ALLFLG and messages occur.

The DEX routine RGET is used to read the real scalar from the database and it returns error codes for either success or the problems which can occur. The latter are brought to the user's attention. DEX routine REALIN (reference [5]) is used to read the real scalar from the terminal.

In cases where an error occurs, the user is informed if the variable is essential for input continuation. When VITAL and ALLFLG are both .TRUE., ALLFLG is set to .FALSE. and the user is informed. Further, if IOMODE=1 but there is no open database, or IOMODE=4 but a premature end-of-file is encountered, ALLFLG is set to .FALSE. regardless of the value of VITAL.

RSREAD is set to .TRUE. if successful and .FALSE. if an error occurs, and control is then returned to the calling program. When successful, prior to returning control, the value read is converted into program standard units by the expression

$$RVAL=RVAL * UNITFM + UNITFA$$

4.4 Real Array Series

4.4.1 Brief description. The real array reading routines include RALDR and RALRED, plus RVAPMP which is shared with the real scalar series. Their function is to read one dimension real arrays, up to the current DEX limit of 200 elements, from any of the four valid sources of input. The reading of x-y coordinates from the screen, while legitimate for an array since it can store a pair of real numbers, has not been implemented yet and the user is advised of this. The next generation of DEX at MIT will include routines for reading and writing two arrays simultaneously for graphics tasks.

4.4.2 RALDR.

4.4.2.1 Calling sequence. Logical function RALDR invokes RALRED to read a real array from the designated source. It also calls RVAPMP as necessary to prepare a prompting message for terminal input. Its calling sequence, listed here, has a few parameters not seen in RSCLDR:


```

LOGICAL FUNCTION RALDDR(RLARR,ALLFLG,NGOT,
                        MTERSE,IOMODE,NCPW,DBNAME,
                        MXTOGT,UNITFM,UNITFA,UNITNM,
                        VITAL
                        PMPREP,PMES,PMORGN,RNFILE,FORMAT,
                        NDEF,DEFAULT)

```

RLARR is the real array that will store the elements, in program standard units, that are read. MXTOGT represents the maximum number of elements to be read into RLARR, and is the dimensioned size of RLARR. NGOT is the number of elements actually read and can be less than or equal to MXTOGT. NGOT need not be defined when RALDDR is invoked. NDEF is the number of default values and is provided to allow the default condition to be smaller than the maximum capability of the array.

4.4.2.2 Characteristics. When IOMODE=2 and PMPREP=.TRUE., RALDDR invokes RVAPMP to prepare PMES. When RVAPMP is successful, and for the other valid sources of input (IOMODE=1, 4 or 5), RALDDR calls RALRED. If either RVAPMP or RALRED are not successful, RALDDR is set equal to .FALSE.. This also occurs when IOMODE=3. It is set to .TRUE. otherwise and control is returned to the calling program.

4.4.3 RALRED

4.4.3.1 Calling sequence. The calling sequence for RALRED is as follows:


```
LOGICAL FUNCTION RAlRED(RlARR,ALLFLG,NGOT  
                        MTERSE,IOMODE,NCPW,DBNAME,MXTOGT,  
                        UNITFM,UNITFA,VITAL,  
                        PMES,RNFILE,FORMAT,NDEF,DEFAULT)
```

Like in RSREAD, UNITNM, PMPREP and PMORGN are no longer required. MXTOGT represents the maximum number of elements to be read into RlARR, always starting at position 1. NGOT is defined in RAlRED.

4.4.3.2 Characteristics. RAlRED first employs DEX routine CHKRNG to verify that MXTOGT is not greater than the maximum DEX array size (currently 200 elements).

If IOMODE=3, the real array is not read and the appropriate checks of VITAL and ALLFLG and message issuing occur.

The reading of an array from a database is accomplished by DEX routine AGET, which seeks MXTOGT elements from the database array. AGET returns six possible result codes. RCODE=0 is simple success, that is, there were MXTOGT elements stored in the database array. NGOT is set equal to MXTOGT. If RCODE=1, there was no open database. This causes ALLFLG to change to .FALSE. if it was .TRUE. and RAlRED to be set to .FALSE.. RAlRED is also set to .FALSE. if the variable does not exist in the database (RCODE=2), if it is not an array (RCODE=3), or if it was undefined (RCODE=4). When the number of

elements requested exceeds the number stored (RCODE=5), the extra elements in R1ARR are set equal to 0.0 but NGOT is set equal to the number stored. When the number of elements requested is less than the number of elements stored (RCODE=6), the first MXTOGT elements are read into R1ARR and NGOT is set equal to MXTOGT. The user is advised in these circumstances of what has occurred.

When reading from the terminal, DEX logical function REALIN is invoked with the following statement:

```
LOGVAL = REALIN (MXTOGT,NEED,R1ARR,PMES)
```

A prompting message asks the user to input up to MXTOGT values. NEED represents the difference between the number of elements read in and MXTOGT. If no elements are read (NEED=MXTOGT) the reading is considered a failure. The reading is considered successful if at least one number is entered. NGOT is set equal to the number of elements entered.

The user is advised if a premature end-of-file is encountered when reading from a file.

When failures occur, VITAL and ALLFLG are processed as usual. Then R1RED is set to .FALSE. and control returns to the calling program. If the reading is successful, the elements are converted into program

standard units by a DO loop for NGOT iterations with the statement

$$R1ARR(I) = R1ARR(I)*UNITFM + UNITFA$$

Then RAlRED is set to .TRUE. and control returns to the calling program.

CHAPTER 5

THE EXTENDED DEX LIBRARY EDITING ROUTINES

5.1 General Description

At some point in the operation of a program the user may decide that he wants to change the value of one or many variables. It may be that the value read as input is incorrect, or he wants to see the effect of changing one variable on the output. He may even decide he does not like the answer given by his program and, before storing it in a database or file, wishes to exchange it for another value he has. For whatever reason, the user requires the capability to enter the new value at the terminal. The editing routines were developed for this purpose.

Currently there are two logical functions in this category, ISCEDT and RSCEDT, with a third RAREDT, scheduled to be developed for the next version of the extended DEX library. ISCEDT allows the editing of integer scalar variables, RSCEDT allows the editing of real scalars, and RAREDT will allow the changing of elements in a real array.

5.2 Logical Function ISCEDT

5.2.1 Calling sequence. Logical function ISCEDT would be invoked by a module subprogram, normally when IOFLAG is set equal to 2 in a subroutine like MAINPG described in Chapter 2. The calling sequence includes 15 parameters listed here:

```
LOGIAL FUNCTION ISCEDT(NEWVAR,ALLFLG,  
                      MTERSE,NCPW,DBNAME,  
                      MENUFL,MENUNM,NITEMS,ITEMS,  
                      PMPREP,PMES,PMORGN,RNFILE,  
                      FORMAT,DEFAULT)
```

For ISCEDT, the first parameter is an output variable, ALLFLG is both input and output, and the remainder are all input variables.

NEWVAR will store the new value of the integer variable being changed. ALLFLG indicates the status of the calling program "all" option. Its value may be changed from .TRUE. to .FALSE. during the editing sequence.

Logical variable MTERSE indicates the type of module dialogue: terse or verbose. NCPW represents the number of characters per word assumed by DEX routines, and is dependent upon the particular computer in use. DBNAME is the 8-character database name of the integer sought.

When MENUFL=.TRUE., the integer being sought is a menu selection. The eight-character menu name is stored

in MENUNM. NITEMS is the number of menu items, and it cannot exceed 12. ITEMS is where the menu choices are stored. Each item is described by an eight-character name (including blanks), so that, at four characters per word, ITEMS should be dimensioned as 2*NITEMS.

Since the new integer value will be entered at the terminal, a prompting message is required. PMPREP is a logical variable which indicates if the program is to prepare the message (.TRUE.) or if it is supplied by the calling program (.FALSE.). PMES is where the prompting message is stored. It can be up to 64 characters long, and if less it must include the trim character, "<", at its end. If PMPREP=.TRUE., PMES is undefined in ISCEDT.

PMORGN stores the information describing the variable in question. It is typically the database comment. PMORGN can be up to 64 characters long and requires the trim character if less. Because PMORGN is used to prepare the prompting message when the dialogue is verbose, if PMPREP=.FALSE., it need not be defined in ISCEDT.

RNFILE is the reference number for reading from a sequential file using Fortran READ and corresponds to RNRFIL in the calling program. FORMAT stores the format for reading from a file. DEFAULT is the default value of the integer in question.

5.2.2 Operation. The task of ISCEDT is actually quite simple. In order to read a new integer from the terminal, ISCEDT merely invokes ISCLDR, using the variable EDMODE, which has a value of 2, in place of IMODE. ISCLDR then prepares a prompting message, if necessary, and calls ISREAD to read the value entered, ISCEDT is set to the same value with which ISCLDR is returned (i.e., .TRUE. for success), and control returns to the calling program.

In calling ISCLDR, ISCEDT defines the parameter VITAL as .TRUE. in all cases. This stems from the policy that if the user wishes to correct a value, he really wants to correct it for program continuation. Failure of any of the integer reading routines would change ALLFLG if it was .TRUE. when ISCEDT was invoked.

It may not be readily apparent to the reader, but because the source will always be defined as the terminal by ISCEDT, the calling parameters RNFILE, FORMAT and DEFAULT, used only when IMODE=4 or 5, need not be defined here. In fact, dummy variables could have been used. It was decided to use the correct variables in the calling sequence to avoid potential errors by creating more variables. The ones used should already be available in the module, being needed for reading and writing integer values.

5.3 Logical Function RSCEDT

5.3.1 Calling parameters. Logical function RSCEDT is invoked by the module to permit the editing of a real scalar variable. Its calling sequence is as follows:

```
LOGICAL FUNCTION RSCEDT(NEWVAR,ALLFLG,  
                        MTERSE,NCPW,DBNAME,  
                        UNITFM,UNITFA,UNITNM,  
                        PMPREP,PMES,PMORG,NRFILE,  
                        FORMAT,DEFAULT)
```

All of the parameters except NEWVAR are input variables, and ALLFLG is both an input and output variable.

Most of these parameters are identical to those used in ISCEDT. Three are new. UNITFM and UNITFA are respectively the multiplicative and additive conversion factors for converting the real scalar read in user i/o units to the value NEWVAR in program standard units. UNITNM is a 12-character version (including blanks) of the user input/output units of the variable, and is used in preparing the prompting message when PMPREP=.TRUE..

5.3.2 Operation. RSCEDT behaves very similarly to ISCEDT. It calls RSCLDR with the variable EDMODE, defined as 2, in lieu of IMODE (terminal source) and VITAL specified as .TRUE.. The real scalar reading routines accept a value entered at the terminal, convert it to program standard units and store it in NEWVAR. If a failure occurs, ALLFLG changes to .FALSE. if it was .TRUE. when RSCEDT was invoked.

RSCEDT is set to the same value as RSCLDR (.TRUE. if NEWVAR is successfully read in, .FALSE. if an error occurred in the reading sequence) and control is returned to the calling program.

One can see that the editing routines are essentially another version of the reading routines. Current plans for the array editor anticipate extending this capability to include reorganizing the entries and inserting new values for whichever element or group of elements is specified by the user. Further, it is hoped that through the editor, it will be possible to execute the calculation subprograms only for those elements changed in order to reduce computing costs. It may be that these options will be operated by the user by means of editing menus also. In short, the goal will be to simulate the editor capability of an interactive system such as CMS at MIT.

CHAPTER 6

THE EXTENDED DEX LIBRARY WRITING ROUTINES

6.1 General Description

Once information has been read, edited or computed, unless it is to be used as input for computations, it is necessary to transmit it to one of three possible destinations: a DEX-created database, the terminal or a sequential file. Further, for our purposes, a distinction shall be made between writing alphanumeric characters on the terminal screen via DEX routines and plotting the information on the screen as a graph. In the current version of DEX at MIT the plotting option is not yet implemented.

The function of the extended DEX library writing routines, described in this chapter, is to permit the transmission of the information to the three valid destinations. Eight logical functions comprise this group of routines, and, like the reading routines, they can be categorized by either type of variable handled or by function. They are listed here by the first method:

<u>Integer Scalar</u>	<u>Real Scalar</u>	<u>Real Array</u>
ISCDMP	RSCDMP	RARDMP
ISDSCR	RVDSCR	
ISRITE	RSRITE	RARITE

Both the real scalar and real array series share RVDSCR.

The top routines, called the "dumpers", serve a function similar to that of the loaders of the reading routines. They screen out requests to perform plotting, call the "descriptors", if necessary, to prepare description messages for identifying the values when writing on the terminal, and invoke the "writers" to do the actual writing to the destination.

One general observation concerning the writing routines which is best made here is that the concept of "essential" variables, which was introduced in the chapter on the reading routines, is not employed. This affects the execution of a calling program all option. The premise is that when writing all the values from a menu, the failure to write one should not prevent the writing of the remainder. The user can go back and analyze why the one was not successful without having to rewrite all of the variables from the menu. The only case where the all option is aborted is where the destination is a database and no database is found open. Rather than getting a string of similar messages announcing this fact, the writing sequence is halted.

6.2 Integer Scalar Series

6.2.1 ISCDMP

6.2.1.1 Calling sequence. Logical function

ISCDMP is the supervisory subroutine for writing integer scalar values and is the subroutine which appears in module subprograms. The calling sequence contains eleven parameters and is listed here:

```
LOGICAL FUNCTION ISCDMP (ALLFLG, MTERSE, IOMODE, NCPW,  
                        IVAR, DBNAME, DMORGN,  
                        DMPREP, DMES, RNFILE, FORMAT)
```

In accordance with DEX practices, the output variables come first in the calling sequence. ALLFLG is both an input and output variable for ISCDMP, being defined at the invocation and capable of having a different value when ISCDMP is returned to the module subprogram. ALLFLG contains the information about the calling program all option.

The remaining parameters are exclusively input variables from the point of view of ISCDMP. MTERSE indicates the type of module dialogue. NCAW is the number of characters per word assumed by the DEX routines. IOMODE corresponds to OMODE in the module calling program and represents the destination of the information to be written. As a reminder, its values are repeated here:

IOMODE=1: a DEX-created database

IOMODE=2: the terminal using DEX routines

IOMODE=3: the screen using plotting routines

IOMODE=4: a sequential file using a formatted
WRITE statement

IVAR is the integer value which is to be written.

DBNAME is the 8-character database name of the variable.

DMORGN is a string of up to 64 characters which describes the variables. It is usually the database comment. If it has less than 64 characters it must include the trim character "<". The routines in this series assume that integer variables have no units.

DMPREP is a logical variable which, if .TRUE., means that the description message is to be prepared by ISDSCR. In this case, DMES is undefined. DMES is where the description message is stored. If the dialogue is verbose it can be up to 64 characters by, whereas if the dialogue is terse it will only contain DBNAME. If DBPREP=.FALSE., DMES must be provided by the module and must include the trim character if it is less than 64 characters long.

RNFILE is the file reference number for writing to a sequential file. It corresponds to RNWFIL in the calling program. FORMAT contains the format to be used to write the integer available to the file.

6.2.1.2 Characteristics. ISCDMP first checks the destination pointer. If IOMODE=3, the user is informed that plotting is an improper mode of output for an integer scalar value and ISCDMP is set to .FALSE. before returning control to the calling program.

If IOMODE=2 and DMPREP=.TRUE., ISCDMP calls ISDSCR to prepare the description message for identifying the integer when it is written to the terminal. When this is successfully accomplished, or when IOMODE=1 or 4, ISCDMP invokes ISRITE to actually perform the writing.

If either ISDSCR or ISRITE is returned .FALSE., ISCDMP is also set to .FALSE. and control is returned to the calling program. Otherwise, it is set to .TRUE. prior to returning control.

When invoking ISRITE, a new logical variable, DBCHNG, is introduced. It is included in anticipation of future capabilities of DEX and indicates when a change is made to a database value. This will alert the user to check other variables which are dependent on the value of the integer being written. Currently DBCHNG is initialized as .FALSE. in ISCDMP.

6.2.2 ISDSCR

6.2.2.1 Calling sequence. Logical function ISDSCR prepares a description message suitable for

identifying the integer available when it is to be written on the terminal. Its calling sequence lists the pertinent parameters provided by ISCDMP:

LOGICAL FUNCTION ISDSCR(DMES,MTERSE,NCPW,DBNAME,DMORGN)

The value of logical variable MTERSE dictates whether the description message, to be stored in DMES, will be brief or long. NCPW is used by the DEX routines which manipulate character strings to produce the message.

6.2.2.2 Characteristics. If the dialogue is verbose, DMORGN, which contains a string of up to 64 characters describing the integer variable in question, is inserted into DMES. For terse dialogue, DBNAME is copied into the description message. If the insertion is successful, ISDSCR is set to .TRUE. and control returns to ISCDMP. If not, an error message is issued and ISDSCR is returned .FALSE..

6.2.3 ISRITE

6.2.3.1. Calling sequence. Logical function ISRITE actually writes the integer available to the specified destination. Its calling sequence is as follows:

LOGICAL FUNCTION ISRITE(ALLFLG,DBCHNG,MTERSE,IOMODE,
NCPW,
IVAR,DBNAME,DMORGN,DMES,RNFILE,
FORMAT)

Logical variables ALLFLG and DBCHNG are defined when ISRITE is invoked. DBCHNG is always .FALSE., and will become .TRUE. if a change is made to the integer variable DBNAME in the database. DMES is always defined when ISRITE is invoked so DMPREP is no longer needed. DMORGN is required because it may be used for comparison with the database comment.

6.2.3.2 Characteristics. If the destination of the integer available is a database, ISRITE first attempts to extract an existing value by DEX routine IGET (reference [5]). If no database is open, ALLFLG is changed to .FALSE. if it was not already, with an appropriate message being issued to the user. If the variable is defined in the database, and it is different from the integer available, both are presented for the user's inspection. The user then specifies which is to be placed in the database. The new value is inserted by DEX routine IPUT (Reference [5]) and DBCHNG becomes .TRUE.. If the variable was not defined, the new value is automatically inserted. Once this is accomplished, the database comment is compared to DMORGN and, if different, they are presented for the user's inspection. Again, he specifies which is to be the final database comment.

When writing to the terminal, an output message is created from DMES, the string " = " and the integer value. The entire message is then printed by DEX routine MESOUT.

If IOMODE=3, the user is informed that plotting is an improper mode of output for an integer scalar. In this case, and other cases where the writing is unsuccessful, ISRITE is returned .FALSE. to the calling program. Otherwise it is set to .TRUE. prior to returning control.

6.3 Real Scalar Series

6.3.1 RSCDMP

6.3.1.1 Calling Sequence. RSCDMP is the subprogram that normally appears in the module for writing a real scalar value to the designated source. Its calling sequence includes 14 parameters:

```
LOGICAL FUNCTION RSCDMP(ALLFLG,MTERSE,IOMODE,NCPW,  
                        RVAR,DBNAME,UNITFM,UNITFA,  
                        UNITNM,  
                        DMPREP,DMES,DMORGN,RNFIL,  
                        FORMAT)
```

All of these parameters are input variables with respect to RSCDMP except ALLFLG, whose value may be changed during the writing sequence. RVAR stores the value of the real scalar available to be written. UNITFM and UNITFA are respectively the multiplicative and additive

conversion factors for converting the real value in program standard units to input/output units prior to the writing. UNITNM is a 12-character version (including blanks) of the input/output unit name. DMPREP indicates if the description message, DMES, is to be prepared by RVDSCR.

DBNAME is an 8-character name of the real scalar and DMORGN is a string of up to 64 characters which identifies the variable. If the variable is dimensioned, DMORGN contains the string "(????????????)", referred to as UNITPT, into which UNITNM will be inserted. DMORGN must include the trim character if it is less than 64 characters long. RNFILE, corresponding to RNWFIL of the module calling program, is the file writing device number. FORMAT contains the format for writing to a file with a formatting Fortran WRITE statement.

6.3.1.2 Characteristics. RSCDMP has three tasks: to screen out requests to plot a real scalar, to call RVDSCR if DMPREP=.TRUE. and IOMODE=2, and to call RSRITE to perform the actual writing. If IOMODE=3, RSCDMP issues a message informing the user that this is not possible. In this case, if either RSDSCR or RSRITE are returned .FALSE., RSCDMP is set to .FALSE. and control is returned to the calling program. If the called

functions are returned .TRUE., RSCDMP is also set to .TRUE. before returning control to the calling program.

In invoking logical function RSRITE, the logical variable DBCHNG is introduced. It is initialized in RSCDMP as .FALSE.. If, when executing RSRITE the variable value is changed in the database, DBCHNG is returned to RSCDMP as .TRUE.. In future versions of DEX, RSCDMP will pass the value of DBCHNG back to the calling program to alert the user to check other variables which are dependent on RVAR.

6.3.2 RVDSCR

6.3.2.1 Calling sequence. In the same manner as RVAPMP, RVDSCR is shared by both the real scalar and real array series. Its function is to prepare a description message suitable for identifying the values being written on the terminal. It is invoked by either RSCDMP or RARDMP using the following calling sequence:

```
LOGICAL FUNCTION RVDSCR(DMES,MTERSE,NCPW,DBNAME,  
                        DMORGN,UNITNM)
```

DMES is undefined when RVDSCR is invoked. The other parameters are all input variables from this function's point of view. They have all been described in either section 6.2.1.1 or 6.3.1.1.

6.3.2.2 Characteristics. If the module dialogue is verbose (MTERSE=.FALSE.), the description message is formed by copying DMORGN into DMES. If the real scalar being written has units, RVDSCR inserts the 12-character unit name UNITNM into the string UNITPT, "(????????????)", which is now in DMES by virtue of having been in DMORGN. If the dialogue is terse, then RVDSCR copies DBNAME into DMES. It then scans DMORGN for UNITPT, and if it finds it, copies UNITPT into DMES following DBNAME and replaces the question marks with UNITNM. If DMES is successfully prepared, RVDSCR is returned .TRUE.. Otherwise it issues an error message, is set to .FALSE., and returns control to the calling program (either RSCDMP or RARDMP).

6.3.3 RSRITE

6.3.3.1 Calling sequence. Logical function RSRITE is used to write a real scalar to the valid specified destination. Its calling sequence is as follows:

```
LOGICAL FUNCTION RSRITE (ALLFLG, DBCHNG, MTERSE, IOMODE, NCPW,  
                        RVAR, DBNAME, UNITFM, UNITFA, UNITNM,  
                        DMORGN, DMES, RNFILE, FORMAT)
```

This is similar to RSCDMP with three exceptions. DBCHNG is a logical variable which is always .FALSE. when RSRITE is invoked. DMPREP is no longer needed since in all

cases DMES is now defined. DMORGN is still required because it will be compared with the database comment.

6.3.3.2 Characteristics. RSRITE first converts the real scalar value from program standard units to user input/output units, stored in variable TVAR, by the statement:

$$\text{TVAR} = (\text{RVAR} - \text{UNITFA}) / \text{UNITFM}$$

If IOMODE=1, the database is first checked to see if a value in question already exists. If the database is found closed, RSRITE alerts the user and changes ALLFLG to .FALSE. if it was not already so. If the variable does not exist in the database, it is inserted using DBNAME, TVAR and DMORGN (corrected for units if applicable) as its name, value and database comment. If the variable exists but is not a real scalar, RSRITE informs the user and is set to .FALSE..

If the variable exists and is defined, its value is compared to TVAR and the user is presented with both if there is a difference. He then is asked which value he wants in the database and the chosen one is written (or left) in. RSRITE then compares the existing database comment to the one specified by DMORGN and writes them both for the user's inspection if they are different. The user then specifies which one is to be the

database comment. This step is crucial in insuring that the correct units for TVAR exist in the database comment.

When writing to the terminal, an output message is constructed using DMES, the string " = " and the real value. This message is then printed by DEX routine MESOUT.

If IOMODE=3 despite the previous checks, the user is informed that a plot cannot be used for writing a real scalar value, and RSRITE is returned .FALSE. . RSRITE is set to .FALSE. in all cases where the real value is not successfully written and control is then returned to the calling program. Otherwise it is returned .TRUE. to RSCDMP.

6.4 Real Array Series

6.4.1 RARDMP

6.4.1.1 Calling sequence. Logical function RARDMP is used in the module subprogram for writing a real array. It has the same three functions as RSCDMP and ISCDMP: to screen out requests to plot graphs, to call RVDSCR if needed to prepare a description message, and to call RARITE to actually do the writing. It has the following calling sequence:

LOGICAL FUNCTION RARDMP (ALLFLG, MTERSE, IOMODE, NCPW,
R1ARR, DBNAME, NFROM, NTO,
UNITFM, UNITFA,
UNITNM, DMPREP, DMES, DMORGN,
RNFILE, FORMAT)

A few new parameters deserve explanation. R1ARR stores the array elements, in program standard units if they have dimensions. The array corresponding to R1ARR in the module should be dimensioned as large as the value MXTOGT used in RALLDR for reading the array.

NFROM represents the position in the array at which writing commences. It should always have a value of 1, specified in the module calling program, except possibly when writing to the terminal. If the user is in the "editing" versus the "writing" mode of operation, he may desire to write only part of an array on the terminal. In this case the editing routine specifies NFROM to be a value from 1 to NTO inclusive prior to invoking RARDMP. NFROM should be 1 when writing the array on the terminal when not in the "editing" mode.

NOT represents the number of elements to be written in all cases but one. This exception occurs when writing to the terminal in "editing" mode, when NTO indicates the last element in the array to be written. Other than this case, NTO corresponds to the value NGOT obtained when reading the array with the reading routines (i.e.,

the actual number of elements read into the array represented by R1ARR), and may be less than MXTOGT.

The other parameters in the calling sequence are the same as those in RSCDMP.

6.4.1.2 Characteristics. If IOMODE=2 and DMPREP=.TRUE., RARDMP invokes RVDSCR to prepare a description message suitable for identifying the array being written to the terminal. If RVDSCR is successful, and for IOMODE=1 and 4, RARDMP invokes RARITE with the statement

```
LOGVAL=RARITE (ALLFLG,DBCHNG,MTERSE,IOMODE,NCPW,  
              R1ARR,DBNAME,NFROM,NT0,UNITFM,UNITFA,  
              UNITNM,  
              DMORGN,DMES,RNFILE,FORMAT)
```

In this version of DEX, DBCHNG is initialized .FALSE. in RARDMP. If a change is made to a database array by RARITE it will be changed to .TRUE.. In future versions RARDMP will pass the value of DBCHNG back to the user via its calling sequence, to alert the user who may wish to verify other variables dependent upon this array.

If IOMODE=3, RARDMP informs the user that it cannot be used to write a real array. In this case, or if RVDSCR or RARITE is returned .FALSE., RARDMP is set to .FALSE. prior to returning control to the calling program. If the two called functions are successful, RARAMP is also set to .TRUE..

6.4.2 RARITE. Since the calling sequence for RARITE has been described above, this section will only discuss RARITE's characteristics. The task of this logical function is to write the real array elements available to the proper specified destination. The first action it takes is to convert the elements in program standard units to input/output units and store them in a temporary array. This is done with a DO loop from NFROM to NTO and the statement

$$RTARR(I) = (RLARR(I) - \text{UNITFA}) / \text{UNITFM}$$

The elements in RTARR are in the units described by UNITNM.

If the destination is a database (IOMODE=1), it is desired to compare the existing array with the new one. RARITE first attempts to extract the existing array and store it in a working array RXARR using DEX routine AGET by the calling sequence

$$\text{LOGVAL} = \text{AGET}(\text{DBNAME}, \text{RXARR}, \text{NTO}, \text{NSTORD}, \text{RCODE})$$

There are six possible result codes returned by AGET. If RCODE=0, AGET was completely successful in that the number of elements stored in the database array (NSTORD) is equal to the number requested (NTO), which is also the number of new elements to be stored. If RCODE=1, the database was not open. This will cause ALLFLG to change to .FALSE. if it was not already, aborting the calling program all option.

If DBNAME does not exist in the database (RCODE=2), it is created with DEX function DBVINS, and RTARR is stored in it. RTARR is also immediately stored if the array DBNAME exists but has no datum stored in it (RCODE=4). If DBNAME is not a real array (RCODE=3), the user is informed.

The final two result codes are more diabolical. If the number of elements stored in the database array is less than the number requested by AGET (RCODE=5), the elements that do exist, plus zeros up to NTO elements, are stored in RXARR. The user is advised that this has occurred, that comparison of the existing values in RXARR and the new values in RTARR can be accomplished, but that the new values cannot be stored if the user decides they are the ones desired. This is because the storing of an array is performed by DEX routine APUT via the statement

```
LOGVAL=APUT(DBNAME,RTARR,NT0,NSTORD,RCODE)
```

Unless NTO=NSTORD, the storing will not occur. All is not lost, however! The user can proceed back to the module calling program, exit to the DEX level via the "\$" command and delete the array with the DEX editing capability. He can then return to the module and write the array into the database when AGET returns RCODE=2.

The other problem occurs when the number requested is less than the number stored (RCODE=6). In this case NTO elements are stored in RXARR for comparison, but for the same reason as above, the user is advised that storing the new values will not be possible.

Once RXARR is established (RCODE=0, 5 or 6), a comparison between its elements and those of RTARR is conducted. The criteria for difference is 1.0×10^{-6} . The user is informed of how many differences were found and asked if an inspection of all the values is desired. A partial review is not possible. If the user responds affirmatively, the values are listed. UNITNM is printed in the heading. RARITE then asks the user to specify which group of values (all old or all new) is desired. If the user chooses to insert the new values, DBCHNG is set to .TRUE. and APUT is called to store the values. Error messages will be issued if NTO does not equal NSTORD.

If the writing is successful, or if the old values are retained, RARITE proceeds to compare the database comment to DMORGN, corrected for units, if applicable. When not the same, it prints both and asks the user to decide which is correct, storing the one chosen as the database comment. If there was no comment already in the database, DMORGN is automatically inserted.

When the destination is the database, the writing is considered successful only when all the new values are successfully stored or the old values retained. All the other possibilities result in RARITE being returned to RARDMP as .FALSE..

If IOMODE=2, the description message, DMES, is printed on the terminal, and then the array is listed from position NFROM to NTO. If IOMODE=3, the user is informed that plotting the array cannot be accomplished and RARITE is set to .FALSE..

When IOMODE=4, the array is written to a sequential file by a DO loop from 1 to NTO with the statement

```
WRITE(RNFILE,FORMAT) NTO,(RTARR(I),I=1,NTO)
```

In the cases where the writing is successful RARITE is set to .TRUE. and control is returned to the calling program.

CHAPTER 7

THE EXTENDED DEX LIBRARY UNIT ROUTINES

7.1 General Description

The module author will invariably write the computational subprograms of the module in the unit system with which he is most familiar. Frequently, it is not the system that the user of the module prefers. The tenet of the DEX philosophy to make modules convenient to use dictated that this problem be addressed. The result was the development of a group of subprograms in the extended DEX library which allow the user to choose from a reasonable selection, the units for input and output purposes for five basic types of measurement, plane angle, force, length, temperature, and time - and combinations thereof.

The twenty-two extended DEX library unit routines can be divided into three categories:

- (i) Five subroutines which enable the user to choose from the options available the preferred input/output (i/o) units.
- (ii) Five logical functions which enable the module to obtain conversion factors which convert the five basic user-specified (i/o) units into the program standard units (p.s.u.) and to obtain the unit names of the i/o units for use in prompting and des-

cription messages on the terminal and database comments.

- (iii) Twelve logical functions which enable the module to obtain the conversion factors and unit names or special names for combinations of the basic units.

This chapter will examine each category.

7.2 The I/O Unit Specifiers

7.2.1 General Description. The extended DEX library includes five subroutines which enable the user to read, edit, or write the five basic units he wishes to use for input and output. These are listed here:

AUNIT	(plane angle)
FUNIT	(force)
LUNIT	(length)
TPUNIT	(temperature)
TUNIT	(time)

The user must choose from the units offered by the particular subroutine menu options. These choices were included in anticipation of the possible needs of most users. Table 7-1 lists the choices available.

7.2.2 Characteristics of a Typical Subroutine.

In execution, the five subroutines simply call ISCLDR to read the input/output unit indicator, ISCEDT to edit the i/o unit indicator, or ISCDMP to write the i/o unit indicator. Because all five subroutines are structured identically, only one, AUNIT, will be described in detail.

Table 7-1. I/O Unit Specifier Subroutines,
Menu Names and Units Available

	AUNIT	FUNIT	LUNIT	TPUNIT	TUNIT
	ANG.UNIT	FOR.UNIT	LEN.UNIT	TEMPUNIT	TIMEUNIT
1	cycle	poundal	inch	Celsius deg.	second
2	radian	poundforce	foot	Fahrenheit deg.	minute
3	degree (ang)	short ton	statute mile	Kelvin deg.	hour
4	minute (ang)	long ton	nautical mi.	Rankine deg.	week
5	second (ang)	dyne	millimeter		month (30 day)
6		newton	centimeter		year (360 day)
7		kilopond	meter		
8			kilometer		

Its calling sequence is listed here:

```
SUBROUTINE AUNIT(UIOAUN,CALALL,IOFLAG,IOMODE,  
                MTERSE,DBAUNN,DBAUNC,PMPREP,  
                PMES,RNFILE,AUNFRM,DEFAUN)
```

The calling sequences for the others are similar.

Table 7-2 lists the comparable distinctive parameters.

UIOAUN denotes the i/o angle unit and can be either an output variable (when reading or editing) or an input variable (when writing). It has the following integer values depending on the specific i/o angle unit:

```
1: cycle  
2: radian  
3: degree (angular)  
4: minute (angular)  
5: second (angular)
```

CALALL is a logical variable which indicates the status of the calling program "all" option. Recall from the Cube Module that subroutine MXUNIT was the calling program for this series. IOFLAG indicates whether the operation is reading, editing, or writing (IOFLAG = 1, 2, or 3 respectively) and dictates whether ISCLDR, ISCEDT or ISCDMP will be invoked. IOMODE indicates the source when reading and the destination when writing. MTERSE, NCPW, PMPREP, PMES, and RNFILE fulfill the same roles as described in previous chapters.

DBAUNN is where the database name of the angle unit, "UIOAUN", is stored. DBAUNC is a character

Table 7-2. I/O Unit Specifier Subroutine Calling Parameters

Subroutine					
Parameter	AUNIT	FUNIT	LUNIT	TPUNIT	TUNIT
I/O Unit Indicator	UIOAUN	UIOFUN	UIOLUN	UIOTPU	UIOTUN
Database name	DBAUNN	DBFUNN	DBLUNN	DBTPUN	DBTUNN
Database comment	DBAUNC	DBFUNC	DBLUNC	DBTPUC	DBTUNC
File format	AUNFRM	FUNFRM	LUNFRM	TPUFRM	TUNFRM
Default variable	DEFAUN	DEFFUN	DEFLUN	DEFTPU	DEFTUN

Table 7-3. Basic Unit Series Calling Parameters

Subroutine					
Parameter	AUNIT	FUNIT	LUNIT	TPUNIT	TUNIT
Conversion factor	CONVA	CONVF	CONVL	CNVTPM CNVTPA	CONVT
One letter abbrev.				NAMTP1	
Two letter abbrev.		NAMF02	NAML02		NAMT02
Three letter abbr.	NAMA03	NAMF03			NAMT03
Five letter abbrev.				NAMTP5	
Six letter abbrev.	NAMA06		NAML06		NAMT06
Eight letter abbr.	NAMA08				
Twelve letter abb.	NAMA12	NAMF12	NAML12	NMTP12	NAMT12
Program standard unit indicator	PSTAUN	PSTFUN	PSTLUN	PSTPUN	PSTTUN
I/O unit indicator	UIOAUN	UIOFUN	UIOLUN	UIOTPU	UIOTUN

string which identifies the angle unit variable. It is used as the database comment and in the preparation of the prompting and description messages. AUNFRM is the format to be used if the angle unit indicator is to be read from or written to a sequential file. DEFAUN is the default value of the angle unit indicator if that is chosen as the source.

In operation, AUNIT branches depending on the value of IOFLAG and calls ISCLDR, ISCEDT, and ISCDMP. When the user wishes to read the angle unit, AUNIT provides menu "ANG.UNIT" with its five choices to ISCLDR. This is an example of when a menu is used to input an integer value. The reader should understand that this is not the name of the unit which is read or written by these subroutines, but rather an integer value which denotes the i/o unit to be used.

7.3 The Basic Unit Series

7.3.1 Series Description. Since the module author knows and provides indicators for the units in which he has written his program, once the user specifies the units he wishes to use during input and output, it is possible to determine the conversion factors for relating the i/o units to the program

standard units. These conversion factors can then be passed on to the loaders when reading variables (real scalars or arrays) to convert their values in i/o units to p.s.u. for the module computing subprograms. Similarly, these factors can be passed on to the dumpers when writing variables to convert their values in p.s.u. to i/o units. The determination of the conversion factors for the five basic types of units is accomplished by five logical functions listed here:

UNITAF	(angle units)
UNITFF	(force units)
UNITLF	(length units)
UNITMP	(temperature units)
UNITTF	(time units)

These functions accomplish one other task: they prepare various alphabetic character versions of the input/output unit names, up to twelve characters long, which are used in database comments and prompting and description messages. This is explained further below.

7.3.2 Calling Parameters. Once again, because all five subroutines are essentially structured the same, only one, UNITFF, will be described in detail. The calling sequence for UNITFF, as it would appear in a module subprogram for reading, editing, or writing input/output variables, is as follows:

```
LOGVAL=UNITFF(CONVF,NAMF02,NAMF03,NAMF12,
```


ALLFLG,PSTFUN,UIOFUN,NCPW)

The sequence is similar for all five functions with two exceptions. The number of versions of the unit names for some is different and UNITMP includes two conversion factors instead of one. Table 7-3 lists the comparable calling parameters for the five functions.

The first four calling parameters are defined by UNITFF and the four are input variables to the function. CONVF is the multiplicative conversion factor which partially converts the force values from i/o units to p.s.u. when reading or editing and does the reverse when writing. The conversion also requires an additive conversion factor which, in all cases except with temperature units, is equal to zero and is provided to the loader, editor, or dumper by the module. The conversion that takes place in the reading routines is of the form:

$$\text{VARIABLE(p.s.u.)} = \text{VARIABLE(i/o unit)} * \text{UNITFM} + \text{UNITFA}$$

where UNITFM is the multiplicative conversion factor determined by one of these functions and UNITFA is the additive conversion factor.

NAMF02, NAMF03, and NAMF12 are respectively two-, three-, and twelve- character abbreviations of the

force unit used during input and output. NAMF12 is used as UNITNM in prompting and description messages and database comments for force variables (recall that UNITNM must be a twelve character version of the relevant unit). Tables B-1 through B-5 in Appendix B list the various abbreviations of the five basic units.

PSTFUN and UIOFUN denote the program standard force unit and the input/output force unit respectively. They can each be an integer between 1 and 7 inclusive, corresponding to the seven permissible force units listed in Table 7-1.

7.3.3 Execution. When invoked, UNITFF calls routine CHKRNG to verify that PSTFUN and UIOFUN are within the permissible range 1-7. UNITFF then uses the pair (PSTFUN, UIOFUN) as an index to a data table included within the function to locate the conversion factor appropriate for converting an input value in the i/o force unit denoted by UIOFUN to the program standard force unit denoted by PSTFUN.

UIOFUN is also used as an index to another data table in the function which contains the various abbreviations of the seven force units. UNITFF employs DEX routine LMOVEC to copy the characters from the data table into the strings NAMF02, NAMF03, and NAMF12.

If a failure occurs in defining either the force unit conversion factor or the unit name, the user is informed that the appropriate variable has not been defined, is essential for i/o continuation, and must be corrected before continuing. ALLFLG is changed to .FALSE. if it was .TRUE. and UNITFF is set to .FALSE.. If successful in accomplishing both tasks it is set to .TRUE..

7.4 Derived I/O Unit Series

7.4.1 Series Description. The third series in the units category contains twelve logical functions for defining conversion factors and unit names for units of measurement formed by combining basic units. These are listed in Table 7-4.

In order to operate these functions, the module author must first have either specified or allowed the user to specify the basic i/o units which are building blocks for these derived unit functions. The module program must then have used the appropriate basic unit series function or functions to obtain the various multiplicative conversion factors and abbreviations. These are then used as calling parameters for the derived units in this series.

Table 7-4. Derived I/O Unit Series

<u>Function</u>	<u>Type of Measurement</u>	<u>Units of Measurement</u>
UAACC	angular acceleration	plane angle/(time) ²
UACCEL	linear acceleration	length/(time) ²
UAREA	area	(length) ²
UFREQ	frequency	plane angle/time
UKVISC	kinematic viscosity	(length) ² /time
UMASS	mass	force-(time) ² /length
UMPOWR	mechanical power	force-length/time
UPRESS	pressure	force/(length) ²
UPSPEC	power spectrum	(length) ² -time
URHO	mass density	force-(time) ² /(length) ⁴
USPEED	speed	length/time
UVOL	volume	(length) ³

There is considerably more diversity in the calling sequences of the twelve functions. Appendix B, Table B-6, lists them for reference. In these functions only multiplicative conversion factors are used to determine the combined conversion factors because none involve temperature units. It should, therefore, be easy to identify CONVA, CONVF, CONVL, and CONVT as the angular, force, length, and time multiplicative conversion factors. The abbreviations of the basic units used in the calling sequences were shown in Tables B-1 through B-5.

One of the functions, UPRESS, will be described in more detail as an example of how they operate.

7.4.2 UPRESS Calling Parameters. UPRESS allows its users to define the unit conversion factor and name for a variable that has the units of pressure (force/area). The calling sequence for UPRESS is as follows:

```
LOGICAL FUNCTION UPRESS(UFPRESS,UNPRESS,ALLFLG,  
                        CONVF,CONVL,NAMFØ3,  
                        NAMFØ2,NCPW)
```

The pressure conversion factor UFPRESS converts the input/output pressure unit to the program standard pressure unit by multiplication when reading or editing and converts the p.s. pressure unit to i/o pressure unit when writing by division. The unit name UNPRES

is used to identify the units of the variable in question for messages and the database comment. UNPRES is a twelve-character string (including blanks). ALLFLG indicates the calling program "all" option. NAMFØ3 is a three-character force unit abbreviation and NAMLØ2 is a two-character length unit abbreviation.

7.4.3 UPRESS Operation. UPRESS first defines the pressure unit conversion factor by the statement

UFPRES = CONVF/CONVL**2

In order to form the pressure unit name, UPRESS defines a twelve-character dummy name variable UXPRES printed here:

" _ _ _ / _ _ * * 2 _ _ _ "

UPRESS inserts, via DEX routine LMOVEC, NAMFØ3 into the first three blank spaces and NAMLØ2 into the fifth and sixth spaces. The three "words" (four characters per word) of UXPRES are then set equal to the three words of UNPRES. As an example, if the force unit was poundforce and the length unit was inches, the final version of UNPRES would be

"LBF/IN**2 "

If a failure occurs in preparing the unit name, a message advises the user and informs him that the problem must be corrected, because it is essential for input/

output continuation. If ALLFLG was .TRUE. it is set equal to .FALSE. and the user is informed that the "all" option is aborted. UPRESS is then set equal to .FALSE. If it is successful, UPRESS is set equal to .TRUE..

Certain combinations of basic units have special universally recognized names used to identity the measurement unit. Where possible, the logical functions provide these names rather than creating a name by its constituents, such as UNPRES was formed in the above example. Table 7-5 lists these special names.

Although there are only twelve types of measurements listed the derived unit series have more versatility than first meets the eye. They can be used for units that have different names but the same basic units. For example, UPRESS can be used for stress units as well as pressure. In addition, they can be used for units that have different basic units but the same format. An example is provided by UAACC and UACCEL, which could be used for any unit type requiring one basic unit in the numerator and a basic unit squared in the denominator. The module author must be careful to supply the correct special parameters in the function calling sequence in the module calling subprogram.

Table 7-5. Special Unit Names

<u>Function</u>	<u>Special Name</u>	<u>Meaning</u>	<u>Occurrence</u>
UFREQ	hertz	cycle/second	UIOAUN=1 and UIOTUN=1
UKVISC	stoke	centimeter ² /second	UIOLUN=6 and UIOTUN=1
UMASS	slug	lbf-second ² /foot	UIOFUN=2 and UIOLUN=2 and UIOTUN=1
UMASS	kilogram	newton-sec ² /meter	UIOFUN=6 and UIOLUN=7 and UIOTUN=1
UMPOWR	watt	newton-meter/sec	UIOFUN=6 and UIOLUN=7 and UIOTUN=1
URHO	slug/ft ³	slug/foot ³	UIOFUN=2 and UIOLUN=2 and UIOTUN=1
URHO	kg/m ³	kilogram/meter ³	UIOFUN=6 and UIOLUN=7 and UIOTUN=1
USPEED	knot	naut. mi./hour	UIOLUN=4 and UIOTUN=3

CHAPTER 8

DEVELOPMENT OF A CRUISER-DESTROYER DATABANK AT M.I.T.

8.1 Considerations in Database Design

8.1.1 Function. When designing a database, the developer must not only consider for what immediate function it is intended, but must also try and anticipate other future demands and organize it accordingly. One solution to this problem, in a sense an avoidance of it, is to create very specialized databases containing information about only one aspect of the overall project involved. The project has a databank comprised of many databases. Physical limitations on the database size, such as the limit of 200 variables in a DEX-created database, suggest this practice. These smaller databases may be more efficient from the point of view of computer costs when it comes to manipulating them. However, the situation can arise where a computer program requires as input data from several different databases, entailing the time consuming effort of opening and closing them all. Only experience in using the databases can reveal the deficiencies in their design.

The function of the cruiser-destroyer databases developed and/or envisioned in the Department of Ocean

Engineering at MIT is to support the naval architect during the concept and preliminary design phases of a ship design. During these phases a variety of products are developed, including the overall vessel dimensions and hull definition, hydrostatic and Bonjean curves, weight and volume estimates, longitudinal weight distribution, propulsion and electrical powering requirements, transverse stability and floodable length checks and general arrangements. The tasks to produce several of these, notably the determination of ship dimensions, weight and volume estimates, powering requirements and transverse stability, can be accomplished with the aid of a computer synthesis model. The REED Model [6] used at MIT is an excellent example of this design tool, and it was the anticipated support of that model that strongly influenced the databases designed in this investigation. The naval architect who chooses to use a synthesis model must carefully determine his input if he desires to use the model efficiently. Being able to draw upon a supply of existing ship information is invaluable to this effort, and this was one of the reasons for developing the cruiser-destroyer databases.

An effective database is one that can be shared by many different engineers involved in the ship design

project, each of whom has a different task to perform. Data should be stored in a form that allows each one to extract the information required and use it directly without having to pass it through some form of interpretation process. An example is a table of offsets database. Ideally, it contains sufficient offsets properly organized such that each one of the programs for hydrostatics, Bonjean curves, cross curves of stability, floodable length, structures and seakeeping can directly access it and obtain the input required without having to go through a "black box" interface program.

The development of a comprehensive computer-aided ship design system that ensures such program/database design requires a "top down" approach to the problem, as described in reference [7]. One starts with the overall objective and works down through functional specifications to complete system design. If successfully accomplished, as a result of strict discipline during the process, no unnecessary capabilities need be developed along the way. One proceeds from each level to the next lower by answering the question of how to provide for the needs of the higher one. This contrasts directly with the traditional method of many individuals writing programs for their specific task, and only after-

wards determining if these programs can be integrated for some higher objective.

8.1.2 Types of Databases. Accepting the concept of a bank of databases to describe a ship, either existing or being designed, we can list the types which will be useful:

1. General description
2. Weights and centers of gravity
3. Longitudinal weight distribution
4. Volumes, areas, and centroids
5. Offsets
6. Equipment specifications and locations
7. Power-speed data
8. Seakeeping data
9. Internal arrangements
10. Topside arrangements

This list is similar to that of the computer-aided ship design system implemented in the Ship Department of the British Ministry of Defense [8].

Storing in a computer databank several of these databases for many classes of ship is extremely helpful as a research resource during the concept design of a new vessel. Taking this one step further, as described in reference [8], is to establish "base" ship databanks made up of all of the database types. If a new vessel is similar to one of these, a copy of the databank provides an excellent starting point to begin defining the new design and can save much redundant work. This is

predicated on the assumption that all the databases of a particular type for all ships are identical in structure and differ only in content. Such a practice is essential to the efficient use of the databanks.

8.2 Organization of the MIT Cruiser-Destroyer Databases

8.2.1 General Databases. During this investigation work was conducted to establish the first two types of databases listed in Section 8.1.2 for eleven classes of U.S. cruisers, destroyers, and frigates. These classes are as follows:

FF-1040	DD-931	CG-16
FF-1052	DD-963	CG-26
FFG-1	DDG-2	CG-47
FFG-7	DDG-40	

This section will describe the organization of the general databases and the next section will describe the weights and centers of gravity databases.

The general databases are so named because they provide a general and not-too-detailed description of the ship class which would be useful to a researcher seeking to determine first estimates for a new design. The information was gleaned from various sources in the open literature, and the respective weight and moment reports and booklets of general plans [9,10,11,12].

The database contains eight categories of variables.

These are:

1. Hull characteristics
2. Propulsion and powering
3. Transverse and directional stability
4. Weapons payload
5. Electronics, fire control, and sensors
6. Aviation capability
7. Complement
8. Gross mass properties

Appendix C is an example of a general database. The individual entries are what would appear if one issued the "dump" command from the DEX level with a particular database open. The order of the listing would not be as they appear here because of a "hashing" function built into the DEX which distributes entries to a database in the memory randomly in order to store them more efficiently.

There are actually 78 variables, listed in Table 8-1 which constitute the eight categories. Each one has a number assigned on the left hand margin. These serve as a convenient indicator for the creation of Fortran names for the various variables associated with each element used in a DEX program. For example, in the module MACHWT described later in this chapter, the program names for the default value and comment statement for propulsion plant type (Item #20) are DEF2Ø and

TABLE 8.1

GENERAL SHIP CHARACTERISTICS DATABASE
FOR U.S. NAVY CRUISER-DESTROYERS

<u>NAME</u>	<u>TYPE</u>	<u>COMMENT</u>	<u>UNITS</u>
HULL CHARACTERISTICS			
1. LOA	R	Length overall	feet
2. LBP	R	Length between perpendiculars	feet
3. BEAMDWL	R	Molded beam at design waterline	feet
4. BEAMMAX	R	Maximum beam	feet
5. T	R	Molded draft to keel	feet
6. CP	R	Prismatic coefficient	
7. CX	R	Midship coefficient	
8. CB	R	Block coefficient	
9. CWP	R	Waterplane coefficient	
10. LCB	R	Longitudinal center of buoyancy as a fraction of LBP aft FP	
11. LCF	R	Longitudinal center of flotation as fraction of LBP aft FP	
12. DEPTH1Ø	R	Depth amidships at centerline	feet
13. DRAFTSON	R	Draft of sonar dome	feet
14. DISPMLD	R	Molded displacement	tons
15. DISPTOT	R	Total displacement including appendages	tons
16. WETSURF	R	Wetted surface	sq. feet
17. FOCSL	I	Raised forecastle (Ø = no 1 = yes)	
18.			
19.			
PROPULSION AND POWERING			
20. PPTYP	I	Type of propulsion plant	Reed
21. SHP	R	Total installed shaft horsepower	hp
22. NSHAFT	I	Number of propeller shafts	
23. NE	I	Number of engines	
24. NB	I	Number of boilers	

TABLE 8.1 (cont'd)

NAME	TYPE	COMMENT	UNITS
25. VSUS	R	Maximum continuous sustained speed	knots
26. VEND	R	Endurance speed	knots
27. ENDUR	R	Endurance range	n. mi.
28. PRPTYP	I	Type of propeller (1 = FP 2 = CRP)	
29. DPROP	R	Propeller diameter	feet
30. RPM	R	Propeller rpm at full power	rpm
31. SSEPTYP	I	Type of primary ship service electrical plant	Reed
32. NSSG	I	Number of primary ship service generators	
33. KWSSER	R	Installed primary ship service generator capacity	kw
34. EMETYP	A(2)	Type of secondary or emergency electrical plant	Reed
35. NEMG	A(2)	Number of secondary or emergency generators of each type	
36. KWEMER	R	Installed secondary or emergency generator capacity	kw
37. KWPEMG	A(2)	Capacity per secondary or emergency generator	kw
38.			
39.			
40.			
TRANSVERSE AND DIRECTIONAL STABILITY			
41. GM	R	Metacentric height uncorrected	feet
42. FSUFCOR	R	Free surface correction	feet
43. CI	R	Waterplane moment of inertia coefficient	
44. FINSTABL	I	Fin stabilizers installed (\emptyset = no 1 = yes)	
45.			
46. NRUDDER	I	Number of rudders	
47.			
48.			
49.			
WEAPONS PAYLOAD			
50. TYPGUNS	A(3)	Type of guns	Reed
51. NGUNS	A(3)	Number of guns of each type	
52. TYPMSL	I	Type of missile launchers	Reed
53. NMSL	I	Number of missile launchers	

TABLE 8.1 (cont'd)

	NAME	TYPE	COMMENT	UNITS
54.	TYPCIWS	I	Type of close-in weapon system	Reed
55.	NCIWS	I	Number of close-in weapon systems	
56.	TYBPDMs	I	Type of basic point defense missile system	Reed
57.	NBPDMS	I	Number of basic point defense missile launchers	
58.	TYPTORPL	I	Type of torpedo launchers	Reed
59.	NTORPL	I	Number of torpedo launchers	
60.	TYPASWL	I	Type of ASW launchers	Reed
61.	NASWL	I	Number of ASW launchers	
62.				
63.				
64.				
65.				
ELECTRONICS, FIRE CONTROL AND SENSORS				
66.	TYPSONAR	A(2)	Type of sonar systems	Reed
67.	TYPDOME	I	Type of sonar dome	Reed
68.	TYPSURAD	I	Type of surface search radar	Reed
69.	TYP3DAIR	I	Type of 3-D air search radar	Reed
70.	TYP2DAIR	I	Type of 2-D air search radar	Reed
71.	TYPGRAD	I	Type of gun fire control radars or directors	Reed
72.	NGRAD	I	Number of gun fire control radars or directors	
73.	TYPMRAD	I	Type of missile fire control radars or directors	Reed
74.	NMSLRAD	I	Number of missile fire control radars or directors	
75.	TYPFCSG	I	Type of gun fire control system	Reed
76.	TYPFCSM	I	Type of missile fire control system	Reed
77.	TYPASWFL	I	Type of ASW fire control system	Reed
78.	TYPTDS	I	Type of tactical data system	Reed
79.				
80.				

TABLE 8.1 (cont'd)				UNITS
NAME	TYPE	COMMENT		
AVIATION CAPABILITY				
81. TYPHELO	I	Type of helicopters carried		Reed
82. NHELO	I	Number of helicopters carried		
83. HIFR	I	Helicopter in-flight refueling capability (1=yes 0=no)		
84.				
85.				
COMPLEMENT (ACCOMMODATIONS)				
86. NOFF	I	Number of ship's officers		
87. NCPO	I	Number of chief petty officers in ship's crew		
88. NCREW	I	Number of enlisted in ship's crew		
89. NFLAGOFF	I	Number of officers on flag staff		
90. NENLSTF	I	Number of enlisted on flag staff		
91. NTROOPS	I	Number of troops		
92.				
GROSS MASS PROPERTIES				
93. TYPMATL	A (2)	Type of material for hull and superstructure respectively		Reed
94. WEIGHT17	A (7)	Weights of weight groups 1-7 respectively		tons
95. VCG17	A (7)	Vertical centers of gravity of weight groups 1-7 respectively		feet
96. WTLOADS	R	Weight of Group 8 loads		tons
97. VCGLOADS	R	Vertical center of gravity for Group 8 loads		feet
98.				
99.				
100.				

CMNT2Ø respectively.

In each category some space has been left for additional variables. Further, experience with the databases may indicate that some items are not needed and can be deleted.

An inspection of both Table 8-1 and Appendix C reveals that certain items referring to the types of plant or type of equipment have integer values where one would expect a name. The reason for this is because only three types of variables are allowed in the DEX: integer scalar, real scalar, and real array. Alpha-numeric words in the "value" part of a database entry are not allowed. A code of integer values was needed to solve this dilemma, and it was decided to adopt the payload shopping list of the REED model because of its comprehensiveness and its widespread use at MIT. Appendix D contains the payload list from reference [6], with some additional items included for this application.

The restriction on arrays that they contain only real values poses a minor problem because they sometimes contain information from the code which should be stored as an integer. It should be obvious to the user from the array name that an integer value is implied. Arrays are used in some not very obvious cases in order to

accommodate the most information. An explanation of the array variables should prove helpful.

TYPMATL has two entries to distinguish between the type of material for the hull and the type of material for the superstructure. The integer values are 1 for steel and 2 for aluminum.

The type of sonar carried (TYPSONAR) is an array because some ships have two systems installed: a bow or keel-mounted sonar, plus a towed array or variable depth sonar.

NGUNS and TYPGUNS are three element arrays to accommodate the most number of distinguishable gun mounts in any of the classes, which exists on the DD-931 class. Not only does this destroyer carry two calibers, 5" and 3", but the REED payload code allows the distinction between a 5" gun mounted on the main-deck (93) and a 5" gun mounted on the 01 level (94).

The emergency or secondary electrical plant includes three array variables: EMETYP, NEMG, and KWPEMG. The CG-26 class cruiser has both a gas turbine-driven and diesel-driven emergency generator. Therefore, the first entries of the three arrays describes the one and the second entries describe the other.

Unfortunately, a great amount of the data available

from the various sources for the general database was conflicting. Where such discrepancies occurred, this investigator made choices based upon the most original source, or the value upon which the most sources agreed. Whenever possible, the original ship equipment is listed in order to correspond to the weights and centers of gravity databases, whose information comes from the original class weight and moment reports.

Any value that was either classified or unavailable was left undefined.

8.2.2 Mass Properties Databases. The general databases include a gross mass properties category which includes two arrays, WEIGHT17 and VCG17. These contain respectively the overall weights and centers of gravity of weight groups 1 through 7. The weight groups conform to the U.S. Navy BSCI organization of ship weights. Although the BSCI system has been replaced by the SWBS (Ship Work Breakdown System) in recent years, it was used for the databases because only the FFG-7 class is sufficiently recent to have its weight and moment report organized with the new system. Further, the REED model is based on BSCI.

The gross mass properties are included in the general databases because they are more frequently used for

estimations than the individual weight items, and their inclusion may save the user from inspecting two different databases.

There are about 150 items comprising the eight weight groups of the BSCI system. Therefore, the combination of weight and center of gravity for each item exceeds the limit of 200 entries in a DEX database. Although the use of arrays offers an apparent solution to this problem, the idea was discarded after careful consideration for several reasons. First, if one wished to store the weight in long tons, the vertical center of gravity in feet above baseline and the longitudinal center of gravity in feet aft of the forward perpendicular or from amidships in a three element array, not only would it be difficult to identify the information in the 64 characters of the database comment, but only one of the two unit names could be stored there. Another possibility was to store all of the weights (or centers of gravity) for one weight group in an array. There would then be eight weight arrays, eight vcg arrays and eight lcg arrays, with the proper units in the database comment. The limit of 200 elements per array would not be a problem because the largest index in any weight group is 51. This was considered

unsatisfactory because it was not felt that the one database comment for the array was sufficient to identify the individual weight items and an extra index would have to be provided to the user. Further, an additional process would have to be developed for extracting the particular weight item out of the array, and avoiding the need to know where a value was stored in the database was one of the driving principles for developing DEX databases to begin with.

Instead, it was decided to create a weight database and a vertical center of gravity database, with each item listed separately. Appendix E illustrates the listing of each type. No need was felt by this investigator for a longitudinal center of gravity database for existing ships. The estimating of the transverse stability of a new ship design can be done effectively using data from existing ships because the vertical locations of most items is restricted to a reasonable degree by physical factors or proven arrangements. The REED model demonstrates that dependable parametric equations can be developed for estimating vertical centers of gravity. However, there is far more flexibility in both theory and practice for the longitudinal locations of many of the same items. Therefore, it is

more difficult to correlate into acceptably accurate parametric equations the information available on lcg's in existing ships. This does not preclude the need for a database containing the longitudinal weight distribu-
tion of a ship design in order to support longitudinal strength and seakeeping analyses. Nor does it preclude the use of a longitudinal center of gravity database for a new ship in order to support longitudinal stability (i.e. trim) analyses.

8.3 Independent and Dependent Variables

8.3.1 Concept. Databases can be both the source and destination of information. A particular program may read its input from a database, calculate values for other variables in the database, and write the new values into those entries. This would be disastrous if uncontrolled. When administering a ship design project that involves multiple uses of the same databases, the ship design manager must have a system whereby he can control changes to the databases that occur as the design progresses around the design spiral. Further, the system should allow all design team members to be alerted to changes which may affect them. It is planned in future versions of DEX to implement a system that supports the concept of independent and dependent

variables.

Certain variables will be defined by the user as independent variables which, either by fact or intention, can not be changed despite changes in other variables. The remaining variables in the program or database are dependent on the former or each other for their values. Each entry in a database will be provided with an index of those variables whose value would be affected by a change in its value. When causing a change to such an entry (i.e. DBCHNG becomes .TRUE.), the user can query this index to determine which other items should be checked.

This task is extremely difficult in ship design because of the interaction of almost all of the variables. Ship design is not a linear process but a spiraling one. Figure 8-1 illustrates an attempt to group the variables of the general database into five levels of dependence. The first column represents those variables which can be considered independent. These might appear as specifications in a Top Level Requirement or they might be the result of trade-off studies during the design phase.

The second group consists of those variables which are most directly affected by the independents or which

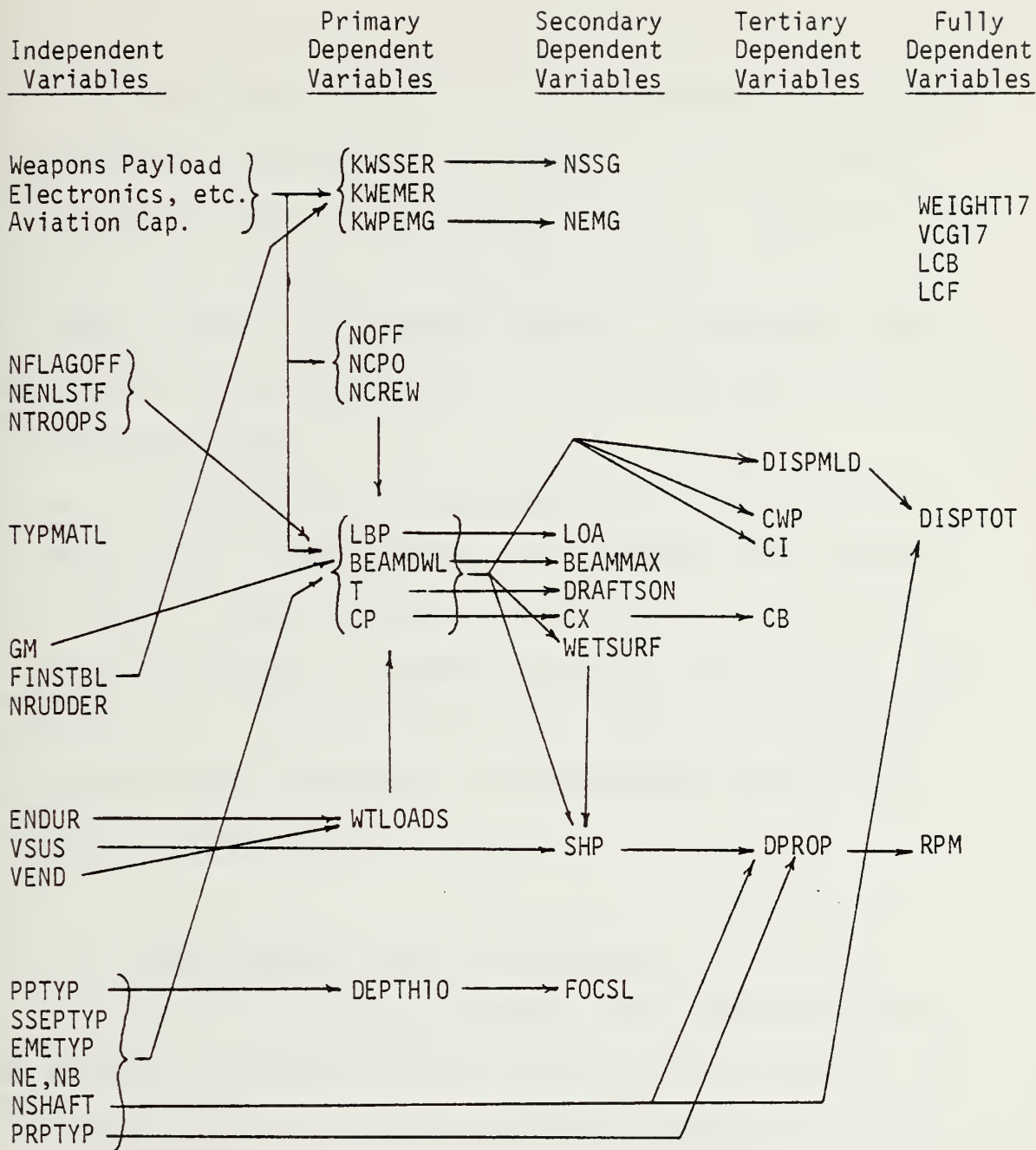


Figure 8-1. General Database Variable Relationships

are estimated first in the design process. The third column is dependent upon values in the first and/or second columns and the fourth column on values in the third and possibly first and/or second column. This table allows the database designer to determine what indices to put on each variable to alert him to check dependent ones.

For example, the dependent variables entry for ENDUR may include the following: WTLOAD, LBP, BEAMDWL, T, CP. A check on LBP will then add to the list of affected variables DISPM LD, DISPTOT, LOA, CWP, CI, LCB, LCF, etc. Although this system requires more work by the database designer, it will make the job of the design manager easier.

8.4 Application of DEX: An Example

8.4.1 Function of MACHWT. The Machinery Weight Estimating (MACHWT) Module was written to demonstrate how DEX and the cruiser-destroyer databases could be used in the preliminary design of a new ship. MACHWT has a fairly limited computation capability since it is only a demonstration module. It enables the user to estimate weight items 200, 201 and 203 based on certain existing parametric equations and parametric equations

developed by the user during the module execution.

These three weight groups are respectively the weight of boilers, weight of propulsion units and weight of the propeller, shafting, and bearings. An analysis of 9 ships for which weight data is available reveals that the sum of these three items constitute between 58.0 and 65.5% of the total Group 2 weight.

The first two weight items are estimated by assuming that they are linearly related to installed horsepower. The program fits a straight line to data extracted from the databases chosen by the user and predicts the new ship weights based on the new specified installed SHP. The program calculates the three component weights of item 203 from the input supplied by the user from any of the valid sources, using parametric equations from the REED model. A summary of the input required for each weight is provided in Table 8-2 (the actual database names are used).

8.4.2 List of Subprograms and Menus. The Machinery Weight Estimating Module includes ten subprograms. They are listed below in the order in which they are most

likely encountered during the execution of the module:

```
MAINPG  
MODIO  
INPUT  
MWUNIT  
MWLIST  
MWCHRT  
MWCOMP  
OUTFUT  
MWCOEF  
BLOCK DATA  
LINFIT
```

There is actually no one correct sequence of listing the subprograms in the module, other than the requirement that MAINPG be first.

TABLE 8-2

INPUT FOR MACHWT

- W200: W200 and SHP from at least two steam ships and SHP of new ship
- W201: W201 and SHP from at least two ships and SHP of new ship
- W203: LBP, PPTYP, NSHAFT, PRPTYP, VSUS and DPROP (optional) of new ship

Seven of the subprograms employ menus in their operation. These are illustrated in Figure 8-2. A listing of the module subprograms appears as Appendix F. They are described in the next section.

8.4.3 Description of the Subprograms. A description of a typical execution of the module will serve as a backdrop for the subprogram descriptions. The user leaves the DEX level and activates the module by using the "DEX-MAIN" menu item and module labeled

.begin machwt

Subprograms MAINPG and menu "MOD.MAIN" are encountered first. MAINPG is identical to the subprogram of the same name used in the Cube Module described in Chapter 2, as is subprogram MODIO, which would be the next one encountered. The menu selections from these two subprograms are

.read input

These place the user in subroutine INPUT. This subroutine provides the user with a menu permitting him to read, edit, or write the following:

1. All the module input variables.
2. The module input and/or output variables
3. The machinery weight item to be estimated
4. The data from existing ships to be used
for curve fitting for weight items
W(200) and W(201).

MENU MOD.MAIN	MENU MOD.IO	MENU INPUT	MENU UNITS
DIALOGUE	INPUT	ALL	ALL
INMODE	OUTPUT	UNITS	FORCE
OUTMODE	DONE	WT.ITEM	LENGTH
READ		CURVEPTS	TIME
EDIT		NEWSHIP	DONE
COMPUT		DONE	
WRITE			
QUIT			

MENU WT.ITEM	MENU CHARACT.	MENU OUTPUT
W200	LBP	UNITS
W201	DRAFT	WT.ITEMS
W203	PPTYPE	COEFFICI
	SHP	DONE
	MAXSPEED	
	NO.SHAFT	
	TYPSCREW	
	DIAM.PRP	
	W200	
	W201	
	W203	

Figure 8-2. Machinery Weight Estimating Menus

5. The characteristics of the new ship design needed as input for the weight calculations.

The machinery weight item must be read first to establish the proper value of a variable WFLAG needed by the subsequent subprograms. This will permit the correct prompting messages to be issued to the user for proper input sequencing.

The user can access MWUNIT to specify the length, force and time units to be used for input and output, but he will normally just use the ones initialized in the module in BLOCK DATA. These values are respectively foot, long ton and second, and were chosen to conform with the units of the database variables used. MWUNIT is a shortened version of MXUNIT from the Cube Module.

Returning from or bypassing MWUNIT, the user types

```
.wt.item w200
```

to access MWLIST and set WFLAG to indicate weight group 200 to be estimated. MWLIST returns him to INPUT and he selects "curvepts". The following prompting message is issued.

```
*SPECIFY THE SEQUENTIAL NUMBER OF THIS PAIR OF DATA POINTS  
*ENTER UP TO 1 INTEGER NUMBERS
```

He types "1" and is presented with menu "CHARACT." from subroutine MWCHRT.

Subroutine MWCHT allows the user to read, edit, or write the characteristics of the ship in question listed in the menu in Figure 8-2. For data points, the independent variable must be read first and the dependent variable next. In this case the user specifies SHP and then W200 and they are read from the open ship database and then inserted into the first positions of an independent variable array and a dependent variable respectively. The user then issues

.done done done

to get back to MAINPG. Using the "inmode" menu selection the user closes the open general database for one steam warship and opens the other one. He then types

.read input curvepts 2 shp yes w200

to input the second pair of data points into the two arrays. The "yes" responds to a question posed by MWCHRT to ascertain if the user is employing horsepower or kilowatts to measure SHP.

This process is repeated for as many ship class databases from which the user wishes to read data for curve fitting, up to a limit of 10. For the purpose of demonstration, the three weight items were stored in the general databases so that only one database for each ship would have to be opened. Normally they re-

side in the weight databases.

When the user is satisfied with the data points read, he specifies "newship" from menu "INPUT" which causes the following message to be issued:

```
*TO ESTIMATE W(200) OR W(201) INPUT NEW SHIP SHP.  
*SELECT WHICH CHARACTERISTIC TO READ.
```

The user then selects SHP from menu "CHARACT." to complete the input required. He returns to MAINPG and executes the computing program MWCOMP by the following command:

```
.done done done compute
```

Once it completes its calculation, MWCOMP returns control to MAINPG, which issues its menu prompting message.

In order to first inspect the coefficients of the straight line fitted to the data, the user (after ensuring that the destination is the terminal) types

```
.write output coeffici
```

These commands invoke MODIO, OUTPUT and MWCOEF successively. The last one causes the two element coefficient array to be printed. The two values which appear are the slope and y-intercept of the straight line.

The user then selects "newship" from menu "OUTPUT" and then "w200" from "CHARACT." and the new estimated boiler weight is printed on the terminal. The user can then return to MAINPG, choose the new ship database

as the destination, and write the estimated W(200) into it. Now, in order to estimate W(201), the user must first exit the module via the "quit" selection from "MOD.MAIN" in order to clear the independent and dependent variable arrays. This is unnecessary if he is going to use at least the same number of data points as for W(200). It is also unnecessary for W(203) which does not require curve fitting.

For W(203), subroutine INPUT prompts the user with the following message when "newship" is chosen:

```
*TO ESTIMATE W(203) THE FOLLOWING INFORMATION IS REQUIRED:  
*LBP PPTYPE SHP NSHAFT PRPTYP VSUS DPROP(optional)
```

If DPROP is not specified MWCOMP estimates it.

Simple as it is, MACHWT is more sophisticated than the Cube Module. It is hoped that the listing in Appendix F can serve as a guide to readers preparing a module for use on the DEX.

8.4.4 Results from the MACHWT Module. The module was exercised to estimate W(200) and W(201) for a nominal new ship design having a 40,000 SHP 1200 psi steam plant installed. In order to estimate the weight of boilers, data from the DDG-2, DDG-40, and FF-1052 classes was used. For estimating the weight of the propulsion units, data from the DDG-2, DDG-40, CG-16, CG-26, FF-1052, and

FFG-1 class databases was used.

The REED Model algorithms for the respective weights are as follows:

$$\begin{aligned}W200 &= .00234 * \text{SHP} + 48.09 \\W201 &= .00143 * \text{SHP} + 17.92\end{aligned}$$

The MACHWT Module fits the following equations to the data used:

$$\begin{aligned}W200 &= .002585 * \text{SHP} + 31.94 \\W201 &= .0017665 * \text{SHP} + 6.66\end{aligned}$$

The respective estimated weights for the new ship appear in Table 8-3.

TABLE 8-3
WEIGHT ESTIMATES FOR 40,000 SHP SHIP DESIGN

	Reed Model	MACHWT
W200 (tons)	141.7	135.3
W201 (tons)	75.1	77.3

8.4.5 Future Developments. MACHWT represents a starting point for what is hoped will be a major ship synthesis program incorporating DEX databases and the REED Model. The model as written contains hundreds of parametric equations for estimating weights, volumes, areas and centers of gravity which were derived from

the data available to its author at that time. As new ships are designed by the Navy, say every 4-5 years, a problem arises with respect to incorporating them into the model. It would be a major undertaking to perform the regression analysis for all new equations. Such a task would have questionable merits since it would probably be found that many equations change only slightly, and others that change drastically have insignificant effects on the overall design. Further, the user would still be confined to using equations of a form chosen by some other designer and derived from those ship classes chosen by him, to which the current user may object.

MACHWT demonstrates a program that allows the user to specify the ship data upon which he wishes to perform a regression analysis. There is no reason why the coefficients obtained could not be written into a database which would be accessed by the REED model in order to estimate that weight item. Expanding on this idea, a program could be developed which allows the user to derive his own coefficients for parametric equations for the large, but not all inclusive, set of variables (weights, volumes, etc.) which impact significantly on the ship design. When a new naval ship class design is

finalized, databases could be produced and stored in the design library at MIT. Only after, perhaps, 3-4 designs and 10-15 years would a major revision of the REED model become worthwhile. The cycle could then begin anew. Not only would this approach avoid frequent rewriting of the REED model, but more importantly, it would allow the individual designer much more control over the tool at his disposal. This would greatly support the function of the department to train naval architects.

CHAPTER 9

CONCLUSIONS AND RECOMMENDATIONS

With the completion of the work of this investigation the first truly capable version of DEX at MIT has been implemented. Current plans call for the adaption to DEX of many of the computer programs in the department and the indoctrination of students to the system. These programs cover a wide range of the calculations which occur during the preliminary design phase.

Two areas of the extended DEX library require development. First is the creation of routines for editing real arrays. Several editing capabilities, similar to those of the operating system, are being considered for implementation, possibly operated by the user by means of an editing menu.

The second area is the task of introducing graphics to the DEX at MIT. An idea to develop routines capable of reading or writing a pair of one-dimensional arrays is under consideration as the means for handling plots. One problem that also must be solved is how to allow the plotting of two curves on the same graph on the screen without any intermediate dialogue between program and user. Although some terminals permit both plotting and

dialogue to occur simultaneously on the screen, many do not, and for DEX to be portable it must be suitable for both types of terminals.

For the purpose of performing ship designs at MIT, this writer perceives the most immediate and imperative need to be the development and implementation of programs which will allow the creation of a table of offsets database. Once the hull form can be defined, the existing programs for hydrostatics, cross-curves of stability, floodable length and Bonjeans, adapted to DEX, can be operated using a common offsets database. Actually, with a hull definition database, the door is open for a significant expansion of the use of the computer in the preliminary design phase including seakeeping, general arrangements, longitudinal strength, etc. Therefore, this task is strongly recommended as a fruitful area for further research.

The adoption of the DEX System entails a change in philosophy on the part of the individual author and user. Heretofore, the programmer required the user to learn how to provide the input, to restrict himself to the design path chosen by the author, and to use the units preferred by the author. With DEX, the user should expect some standardization in the means of input,

flexibility in the path to pursue, and choice in the unit system with which to work. It means more work for the module author, but his job is only performed once, while the advantages he can offer by using DEX will be available to countless users.

REFERENCES

1. C. Chrysosostomidis, "Computer-Aided Ship Design Education at the Massachusetts Institute of Technology," Computer Applications in the Automation of Shipyard Operation and Ship Design II, eds. Jacobsen et al., Amsterdam: North-Holland Publishing Company, 1976, pp. 65-71.
2. John B. Woodward, "Computer-Aided Ship Design Education at the University of Michigan," Computer Applications in the Automation of Shipyard Operation and Ship Design II, eds. Jacobsen et. al., Amsterdam: North-Holland Publishing Company, 1976, pp. 73-78.
3. Bertram Herzog, "A Transportable FORTRAN Based Executive System for Computer-Aided Ship Design Education," Computer Applications in the Automation of Shipyard Operation and Ship Design II, eds Jacobsen et al., Amsterdam: North-Holland Publishing Company, 1976, pp. 79-87.
4. Bertram Herzog, Module Programmer's Guide for the Interactive Computing System DEX at the University of Colorado, (Rev. 1978).
5. R. P. Geize and J. Kelley, DEX Module Programmer's Guide, Houston: Gulf Research and Development Center, Offshore Technology Department, 1979 (under revision).
6. Michael Reed, "Ship Synthesis Model for Naval Surface Ships." O.E. and S.M. Thesis, Massachusetts Institute of Technology, 1975.
7. Craig M. Carlson, Robert A. Johnson and F. William Helming, "Computer Aids for Ship Design, Integration and Control." Naval Engineers Journal (April 1980), pp. 73-87.
8. S. J. Holmes, "The Application and Development of Computer Systems for Warship Design." Paper presented at the meeting of the Royal Institute of Naval Architects, Spring 1980.
9. John E. Moore, ed. Jane's Fighting Ships 1974-75. New York: Franklin Watts, Inc., 1974.
10. Jean Labayle Couhat, ed. Combat Fleets of the World 1978/79: Their Ships, Aircraft and Armament. Annapolis: Naval Institute Press, 1978.

REFERENCES (Continued)

11. Samuel L. Morison and John S. Rowe, compilers. The Ships and Aircraft of the U.S. Fleet. 10th ed. Annapolis: Naval Institute Press, 1975.
12. U. S. Department of the Navy, Naval Sea Systems Command, Naval Vessel Register/Ships Data Book, 1 January 1980.

APPENDIX A

CUBE MODULE LISTING


```

C****-----CUBE MODULE SUBPROGRAM-----0000010
SUBROUTINE MAINPG                                0000020
-----SUBPROGRAM DESCRIPTION-----0000030
C SUBROUTINE MAINPG ALLOWS THE USER TO SELECT THE DESIRED PATH IN THE
C EXECUTION OF THIS MODULE. THE CHOICES ARE      0000040
C 1) SET STYLE OF MODULE DIALOGUE                0000050
C 2) SELECT SOURCE OF MODULE INPUT               0000060
C 3) SELECT DESTINATION OF MODULE OUTPUT         0000070
C 4) INTERROGATE THE MODULE FLAGS AND FIND OUT THEIR VALUE 0000080
C 5) ACCESS THE MODULE READING ROUTINES          0000090
C 6) ACCESS THE MODULE EDITING ROUTINES          0000100
C 7) ACCESS THE MODULE COMPUTING ROUTINES        0000110
C 8) ACCESS THE MODULE WRITING ROUTINES         0000120
C 9) RETURN CONTROL TO DEX                      0000130
C-----LABELED COMMON VARIABLES-----0000140
C.....REFNOS INITIALIZED IN BLOCK DATA      0000150
C RNRFIL: FILE REFERENCE NUMBER FOR MODULE FORTRAN READS FROM A 0000160
C RNRFIL: SEQUENTIAL FILE                      0000170
C RNRFIL: FILE REFERENCE NUMBER FOR MODULE FORTRAN WRITES TO A 0000180
C RNRFIL: SEQUENTIAL FILE                      0000190
C.....INOUTF INITIALIZED IN BLOCK DATA      0000200
C IMODE : DENOTES SOURCE OF MODULE INPUT        0000210
C = 1 A DEX CREATED DATABASE                    0000220
C = 2 THE USER EMPLOYING THE TERMINAL TO INPUT ALPHANUMERIC DATA 0000230
C VIA DEX ROUTINES                             0000240
C = 3 THE USER EMPLOYING THE SCREEN TO INPUT X-Y COORDINATES VIA 0000250
DEX ROUTINES                                   0000260
C = 4 A SEQUENTIAL FILE TO INPUT ALPHANUMERIC DATA WITH FORTRAN 0000270
READS                                          0000280
C = 5 THE MODULE DEFAULT DATA.                0000290
C OMODE : DENOTES DESTINATION OF MODULE OUTPUT 0000300
C = 1 A DEX CREATED DATABASE                    0000310
C = 2 THE TERMINAL USING DEX ROUTINES           0000320
C = 3 THE SCREEN USING PLOTTING ROUTINES        0000330
C = 4 A SEQUENTIAL FILE USING FORTRAN WRITES   0000340
C.....DIALGF INITIALIZED IN BLOCK DATA      0000350
C MTERSE: .TRUE. IF THE MODULE DIALOGUE IS TERSE 0000360
.FALSE. IF THE MODULE DIALOGUE IS VERBOSE     0000370
C.....MNCPPW INITIALIZED IN BLOCK DATA      0000380
C NCPW : NUMBER OF CHARACTERS PER WORD ASSUMED BY DEX ROUTINES 0000390
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----0000400
C DEX                                          0000410
C MENUIN                                     0000420
C ENDTI                                      0000430
C DEX LIBRARY                               0000440
C DIALOG                                     0000450
C SOURCE                                     0000460
C DESTIN                                    0000470
C MDMODE                                    0000480
0000490

```



```

C MODULI
C MODIO
C COMPUT
C-----PROGRAMMERS AND REPORT-----
C PROGRAM VERSION: 1
C PROGRAM DATE : JUNE 1981
C PROGRAMMERS : C. CHIRYSSOSTOMIDIS AND R. CELOTTO
C REPORT : IDLAS ON HOW TO WRITE DEX MODULES
C : VOLUME AND WEIGHT OF S.W. PARALLELPIPED
C : CUBE MODULE ROUTINES
C REPORT NUMBER : 81-1
C AUTHORS : C. CHIRYSSOSTOMIDIS AND R. CELOTTO
C PUBLISHER : MASSACHUSETTS INSTITUTE OF TECHNOLOGY
C : DEPARTMENT OF OCEAN ENGINEERING
C : DESIGN LABORATORY
C : CAMBRIDGE MASS 02139 , JUNE 1981
C-----
C LABELLED COMMONS FOR SUBROUTINE MAINPG
C
COMMON /MODEND/MODEND
COMMON /REFNOS/ RNRFIL,RNWFIL
COMMON /INOUTF/ IMODE ,OMODE
COMMON /DIALGI/ MIERSE
COMMON /MDNCPW/ NCPW
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
INTEGER RNRFIL,RNWFIL
INTEGER IMODE ,OMODE
INTEGER NCPW
INTEGER ITEM,MENUIN,MENUNM(2),NITEMS,ITLMS(18),MESS(1)
INTEGER IOFLAG
C**** START OF SITE DEPENDENT CODE
INTEGER DVFLNM(12)
INTEGER DBFLNM(20)
C**** END OF SITE DEPENDENT CODE
LOGICAL MIERSE
C
C VARIABLE DATA DEFINITIONS
C ALL VARIABLES IN LABELLED COMMONS ARE INITIALIZED IN BLOCK DATA
C
DATA MESS /4H< /
DATA MENUNM/4IMOD.,4IMAIN/
DATA NITEMS/9/
DATA ITEMS /4IIDIAI,4IIQUC,
1 4IINMO,4IHDE ,
2 4IIQUIM,4IHODE ,
3 4IIMOD.,4IIMODE,
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980

```



```

4      4HIREAD,4H      ,
5      4HEDIT,4H      ,
6      4HICOMP,4HUTE   ,
7      4HWRIT,4HE      ,
8      4HQUIT,4H      /

C      C SELECT AN ITEM FROM MENU MOD.MAIN AND BRANCH ACCORDINGLY
C
50 CONTINUE
ITEM=MENUN(MENUNM,NITEMS,ITEMS,MESS)
GO TO (100,200,300,400,500,600,700,800,900),ITEM

C      C SET STYLE OF MODULE DIALOGUE.
C
100 CONTINUE
CALL DIALOG(MTERSE)
GO TO 50

C      C SELECT SOURCE OF MODULE INPUT.
C
200 CONTINUE
CALL SOURCE( IMODE,DBFLNM,DVFLNM,RNRFIL,MTERSE,NCPW)
GO TO 50

C      C SELECT DESTINATION OF MODULE OUTPUT.
C
300 CONTINUE
CALL DESIN(OMODE,DBFLNM,DVILNM,RNWFIL,MTERSE,NCPW)
GO TO 50

C      C SUPPLY INFORMATION ABOUT THE MODULE FLAGS.
C
400 CONTINUE
CALL MDMODE( IMODE,OMODE,RNRFIL,RNWFIL,MTERSE,
              NCPW)
GO TO 50

C      C ACCESS THE MODULE READING ROUTINES.
C
500 CONTINUE
IOFLAG=1
CALL MODIO( IOFLAG)
GO TO 50

C      C ACCESS THE MODULE EDITING ROUTINES.
C
600 CONTINUE
IOFLAG=2
CALL MODIO( IOFLAG)

```

```

00000990
00001000
00001010
00001020
00001030
00001040
00001050
00001060
00001070
00001080
00001090
00001100
00001110
00001120
00001130
00001140
00001150
00001160
00001170
00001180
00001190
00001200
00001210
00001220
00001230
00001240
00001250
00001260
00001270
00001280
00001290
00001300
00001310
00001320
00001330
00001340
00001350
00001360
00001370
00001380
00001390
00001400
00001410
00001420
00001430
00001440
00001450
00001460
00001470

```


C	GO TO 50	00001480
C	ACCESS THE MODULE COMPUTING ROUTINES.	00001490
C		00001500
	700 CONTINUE	00001510
	CALL COMPUT	00001520
	GO TO 50	00001530
C		00001540
C	ACCESS THE MODULE WRITING ROUTINES.	00001550
C		00001560
	800 CONTINUE	00001570
	IOFLAG=3	00001580
	CALL MODIO(IOFLAG)	00001590
	GO TO 50	00001600
C		00001610
C	RETURN CONTROL TO DEX.	00001620
C		00001630
	900 CONTINUE	00001640
	CALL ENDIT	00001650
	RETURN	00001660
	END	00001670
		00001680


```

C-----CUBE MODULE SUBPROGRAM-----MOD00010
C SUBROUTINE MODIO(IOFLAG)MOD00020
C-----SUBPROGRAM DESCRIPTION-----MOD00030
C SUBROUTINE MODIO ENABLES ITS USERS TO SELECT THE SEGMENT OR SEGMENTS
C OF THE MODULE'S VARIABLES THEY WISH TO READ, EDIT OR WRITE.
C THE CHOICES ARE:
C BOTH INPUT AND OUTPUT MODULE VARIABLES
C INPUT MODULE VARIABLES
C OUTPUT MODULE VARIABLES
C-----INPUT VARIABLES-----MOD00090
C IOFLAG=1 IF THIS SUBROUTINE WAS INVOKED BY ITEM READ OF MENU MOD.MAIN
C 2
C 3
C-----LABELLED COMMON VARIABLES-----MOD00130
C LABELLED COMMON DIALG HAS BEEN DEFINED IN SUBROUTINE MAINPG.
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----MOD00150
C DEXMOD00160
C SIRPAKMOD00170
C LMOVICMOD00180
C MINOINMOD00190
C DEX LIBRARYMOD00200
C NONEMOD00210
C MODULMOD00220
C INPUTMOD00230
C OUTPUTMOD00240
C-----MOD00250
C-----MOD00260
C LABELLED COMMONS
C COMMON /DIALG/ MIERSE
C COMMON /MIDNCPW/ NCPW
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
C INTEGER IOFLAG
C INTEGER MESS(15), LMS, NCPW
C INTEGER ITEM, MENUIN, MENUM(2), NITEMS, ITEMS(8)
C INTEGER READ(2), EDIT(2), WRITE(2)
C LOGICAL MIERSE
C LOGICAL ALLFLG, LOGVAL, LMOVEC
C
C VARIABLE DATA DEFINITIONS
C
C DATA EMS /15/
C DATA MENUM/41MOD., 4110 /

```



```

C READ MODULE INPUT DATA.
C
C 200 CONTINUE
CALL INPUT(ALLFLG,IOFLAG)
IF (.NOT.ALLFLG) GO TO 50
C
C READ MODULE OUTPUT DATA.
C
C 300 CONTINUE
CALL OUTPUT(ALLFLG,IOFLAG)
IF (.NOT.ALLFLG) GO TO 50
C
C RETURN CONTROL TO THE CALLING PROGRAM.
C
C 400 CONTINUE
RETURN
END
MOD00910
MOD00920
MOD00930
MOD00940
MOD00950
MOD00960
MOD00970
MOD00980
MOD00990
MOD01000
MOD01010
MOD01020
MOD01030
MOD01040
MOD01050
MOD01060
MOD01070

```



```

C-----CUBE MODULE SUBPROGRAM-----
C      SUBROUTINE INPUT(CALL,IOFLAG)
C-----
C      SUBPROGRAM DESCRIPTION-----
C      SUBROUTINE INPUT PROVIDES THE USER WITH A MENU FROM WHICH TO
C      CHOOSE WHICH MODULE SEGMENT IT IS DESIRED TO OPERATE NEXT. THE
C      CHOICES ARE:
C      ALL MODULE INPUT VARIABLES
C      THE MODULE UNITS TO BE USED DURING INPUT AND OUTPUT
C      THE MODULE DIMENSIONS
C      THE UNITS MODULE ALLOWS THE USER TO SPECIFY THE LENGTH AND FORCE
C      UNITS TO BE USED DURING INPUT AND OUTPUT. THE DIMENSIONS MODULE
C      ALLOWS THE USER TO READ, EDIT OR WRITE THE CUBE DIMENSIONS: LENGTH,
C      WIDTH AND HEIGHT.
C      IF THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE, THE
C      LOCAL ALL OPTION IS SET TO .TRUE. UPON INVOKING THIS SUBROUTINE.
C      IN THIS CASE THE MENU 'INPUT' IS NOT DEFINED.
C-----SUBPROGRAM ASSUMPTIONS-----
C      NONE YET.
C-----
C-----OUTPUT VARIABLES-----
C      CALL: .TRUE. IF THE INPUT VALUE OF CALL WAS .TRUE. AND NO ERROR
C      OCCURRED WHEN READING OR EDITING AN ESSENTIAL INPUT
C      VARIABLE
C      .FALSE. IF THE INPUT VALUE OF CALL WAS .FALSE. OR AN ERROR
C      OCCURRED WHEN READING OR EDITING AN ESSENTIAL INPUT
C      VARIABLE
C-----INPUT VARIABLES-----
C      CALL: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C      .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE
C      IOFLAG=1 IF THE USER WISHES TO READ THE VARIABLES
C      2      EDIT
C      3      WRITE
C-----
C-----LABELED COMMON VARIABLES-----
C      LABELED COMMON DIALOG HAS BEEN DEFINED IN SUBROUTINE MAINPG.
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----
C      DEX
C      SIRPAK
C      LMOVIC
C      MENUIN
C      DEX LIBRARY
C      NONE
C      MODULE
C      MXUNIT
C      DIMENS
C-----
C      LABELED COMMONS
C

```

```

INP00010
INP00020
INP00030
INP00040
INP00050
INP00060
INP00070
INP00080
INP00090
INP00100
INP00110
INP00120
INP00130
INP00140
INP00150
INP00160
INP00170
INP00180
INP00190
INP00200
INP00210
INP00220
INP00230
INP00240
INP00250
INP00260
INP00270
INP00280
INP00290
INP00300
INP00310
INP00320
INP00330
INP00340
INP00350
INP00360
INP00370
INP00380
INP00390
INP00400
INP00410
INP00420
INP00430
INP00440
INP00450
INP00460
INP00470

```



```

COMMON /DIALOG/ MTERSE
COMMON /MNCPU/ NCPW
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
C
C   INTEGER IOFLAG
C   INTEGER MENUM(2),NITEMS,ITEMS(8),ITEM
C   INTEGER MESS(15),LMS,NCPW
C   INTEGER READ(2),EDIT(2),WRITE(2)
C   LOGICAL CALALL,LOCAL
C   LOGICAL MTERSE,LOGVAL,LMOVEC
C
C VARIABLE DATA DEFINITIONS
C
C
C   DATA LMS/15/
C   DATA MENUM/4HINPUT,4HIT /
C   DATA NITEMS/4/
C   DATA ITEMS/4H11,4H1 ,
C   1      4HUNIT,4HS
C   2      4HIME,4HINSIO,
C   3      4HDONE,4H /
C   DATA LOCAL/./FALSE./
C   DATA READ /4HREAD,4H.< /
C   DATA EDIT /4HEDIT,4H.< /
C   DATA WRITE/4HWRT,4HE.< /
C
C
C   C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C   C NOTE THAT IF THE LOCAL ALL OPTION IS SET IN THIS MANNER, THE MENU
C   C 'INPUT' IS NOT DEFINED BY THE INVOCATION OF THIS SUBROUTINE.
C
C   LOCAL=CALALL
C   IF (LOCAL) GO TO 200
C
C   C PREPARE A PROMPTING MESSAGE FOR MENU 'INPUT' AND THEN PROVIDE THE
C   C MENU TO THE USER.
C
C   IF (MTERSE) GO TO 40
C   CALL STRPAK(MESS,LMS,4HK ,4HSELECT WHICH INPUT VARIABLE SEGMENT
C   1 10<
C   GO TO (10,20,30),IOFLAG
C   10 LOGVAL=LMOVEC(READ,1,6,NCPW,MESS,40,NCPW)
C   GO TO 50
C   20 LOGVAL=LMOVEC(EDIT,1,6,NCPW,MESS,40,NCPW)
C   GO TO 50
C   30 LOGVAL=LMOVEC(WRITE,1,7,NCPW,MESS,40,NCPW)
C   GO TO 50
C   40 CALL STRPAK(MESS,LMS,4HK ,21HWHICH INPUT SEGMENT?4
C
INP00480
INP00490
INP00500
INP00510
INP00520
INP00530
INP00540
INP00550
INP00560
INP00570
INP00580
INP00590
INP00600
INP00610
INP00620
INP00630
INP00640
INP00650
INP00660
INP00670
INP00680
INP00690
INP00700
INP00710
INP00720
INP00730
INP00740
INP00750
INP00760
INP00770
INP00780
INP00790
INP00800
INP00810
INP00820
INP00830
INP00840
INP00850
INP00860
INP00870
INP00880
INP00890
INP00900
INP00910
INP00920
INP00930
INP00940

```



```

50 CONTINUE
   ITEM=MENUN(MENUNM,NITEMS,ITEMS,MESS)
   GO TO (100,200,300,400),ITEM
C
C SET THE INPUT ALL OPTION.
C
100 CONTINUE
   LOCAL=.TRUE.
C
C READ, EDIT OR WRITE THE INPUT/OUTPUT MODULE UNITS.
C
200 CONTINUE
   CALL MXUNIT(LOCAL,IOFLAG)
   IF (LOCAL) GO TO 300
   IF (.NOT.CALL) GO TO 50
   CALL=.FALSE.
   GO TO 400
C
C CALL DIMNS TO READ, EDIT OR WRITE THE CUBE DIMENSIONS.
C
300 CONTINUE
   CALL DIMNS(LOCAL,IOFLAG)
   IF (LOCAL) GO TO 400
   IF (.NOT.CALL) GO TO 50
   CALL=.FALSE.
C
C RETURN CONTROL TO CALLING PROGRAM.
C
400 CONTINUE
   RETURN
   END

```

```

INP00950
INP00960
INP00970
INP00980
INP00990
INP01000
INP01010
INP01020
INP01030
INP01040
INP01050
INP01060
INP01070
INP01080
INP01090
INP01100
INP01110
INP01120
INP01130
INP01140
INP01150
INP01160
INP01170
INP01180
INP01190
INP01200
INP01210
INP01220
INP01230
INP01240
INP01250

```



```

C-----CUBE MODULE SUBPROGRAM-----MXU00010
C      SUBROUTINE MXUNIT(CALALL, IOFLAG)MXU00020
C-----SUBPROGRAM DESCRIPTION-----MXU00030
C SUBROUTINE MXUNIT ALLOWS ITS USERS TO SELECT WHICH MODULE UNIT THEY
C WISH TO READ, EDIT OR WRITE. THE CHOICES ARE:
C ALL MODULE UNITS
C THE LENGTH UNIT
C THE TIME UNIT
C THE FORCE UNIT
C THE PLANE ANGLE UNIT
C THE TEMPERATURE UNIT
C-----SUBPROGRAM ASSUMPTIONS-----MXU000110
C IF THE CALLING PROGRAM ALL OPTION IS ACTIVE, THE LOCAL ALL OPTION IS
C SET TO .TRUE, UPON INVOKING THIS SUBROUTINE. IN THIS CASE THE MENU IS
C NOT DEFINED.
C-----OUTPUT VARIABLES-----MXU000150
C CALALL: .TRUE. IF THE INPUT VALUE OF CALALL WAS .TRUE. AND NO ERROR
C OCCURRED IN READING OR EDITING A MODULE UNIT
C .FALSE. IF THE INPUT VALUE OF CALALL WAS .FALSE. OR AN ERROR
C OCCURRED IN READING OR EDITING A MODULE UNIT
C-----INPUT VARIABLES-----MXU000200
C CALALL: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE
C IOFLAG=1 IF THE USER WISHES TO READ MODULE UNITS
C 2
C 3
C 3
C-----WRITE-----MXU000250
C-----LABELED COMMON VARIABLES-----MXU000260
C LABLID COMMON DIALCF, INOUTF, MDNCPW AND REFNO HAVE BEEN DEFINED IN
C SUBROUTINE MAINPG.
C-----UNITS INITIALIZED IN BLOCK DATA-----MXU000270
C PSITUN= / THE PROGRAM STANDARD LENGTH UNIT IS METER
C UTOLUN: DENOTES THE INPUT/OUTPUT LENGTH UNIT
C 1 IF THE INPUT/OUTPUT LENGTH UNIT IS INCH
C 2
C 3
C 4
C 5
C 6
C 7
C 8
C-----INFO INITIALIZED IN BLOCK DATA-----MXU000300
C DBLUNN: WHERE THE DATABASE NAME OF THE LENGTH UNIT IS STORED
C DBLUNC: WHERE INFORMATION FOR IDENTIFYING THE LENGTH UNIT VARIABLE
C IS STORED
C LUNFRM: FORMAT TO BE USED TO READ OR WRITE THE LENGTH UNIT FROM A
C SEQUENTIAL FILE
C DEFLUN: THE DEFAULT VALUE OF THE LENGTH UNIT

```



```

C SUBPROGRAM BLOCK DATA.
C.....UNITS INITIALIZED IN BLOCK DATA
C PSTFUN= 7 THE PROGRAM STANDARD FORCE UNIT IS KILOPOND
C UTOFUN: DENOTES THE INPUT/OUTPUT FORCE UNIT
C      = 1 IF THE INPUT/OUTPUT FORCE UNIT IS POUNDAL
C      = 2 POUND(FORCE)
C      = 3 SHORT TON
C      = 4 LONG TON
C      = 5 DYNE
C      = 6 NEWTON
C      = 7 KILOPOND
C.....UNITS INITIALIZED IN BLOCK DATA
C DBFUNN: WHERE THE DATABASE NAME OF THE FORCE UNIT IS STORED
C DBFUNC: WHERE INFORMATION FOR IDENTIFYING THE FORCE UNIT VARIABLE
C      IS STORED
C FUNFRM: FORMAT TO BE USED TO READ OR WRITE THE FORCE UNIT TO A SE-
C      QUENTIAL FILE
C DEFFUN: THE DEFAULT VALUE OF THE FORCE UNIT
C.....UNITS INITIALIZED IN BLOCK DATA
C PSTFUN= 1 THE PROGRAM STANDARD TIME UNIT IS SECOND
C UTOFUN: DENOTES THE INPUT/OUTPUT TIME UNIT
C      = 1 IF THE INPUT/OUTPUT TIME UNIT IS SECOND
C      = 2 MINUTE
C      = 3 HOUR
C      = 4 DAY
C      = 5 WEEK
C      = 6 MONTH (30 DAY MONTH)
C      = 7 YEAR (360 DAY YEAR)
C.....UNITS INITIALIZED IN BLOCK DATA
C DBFUNN: WHERE THE DATABASE NAME OF THE TIME UNIT IS STORED
C DBFUNC: WHERE INFORMATION FOR IDENTIFYING THE TIME UNIT VARIABLE IS
C      STORED
C TUNFRM: FORMAT TO BE USED TO READ THE TIME UNIT FROM OR WRITE IT TO A
C      SEQUENTIAL FILE
C DEFTUN: THE DEFAULT VALUE OF THE TIME UNIT
C.....UNITS INITIALIZED IN BLOCK DATA
C PSTAUN= 1 THE PROGRAM STANDARD PLANE ANGLE UNIT IS CYCLE
C UTOAUN: DENOTES THE INPUT/OUTPUT ANGLE UNIT
C      = 1 IF THE INPUT/OUTPUT ANGLE UNIT IS CYCLE
C      = 2 RADIAN
C      = 3 DEGREE (ANG)
C      = 4 MINUTE (ANG)
C      = 5 SECOND (ANG)
C.....UNITS INITIALIZED IN BLOCK DATA
C DBAUNN: WHERE THE DATABASE NAME OF THE ANGLE UNIT IS STORED
C DBAUNC: WHERE INFORMATION FOR IDENTIFYING THE ANGLE UNIT VARIABLE
C      IS STORED

```

```

MXU00480
MXU00490
MXU00500
MXU00510
MXU00520
MXU00530
MXU00540
MXU00550
MXU00560
MXU00570
MXU00580
MXU00590
MXU00600
MXU00610
MXU00620
MXU00630
MXU00640
MXU00650
MXU00660
MXU00670
MXU00680
MXU00690
MXU00700
MXU00710
MXU00720
MXU00730
MXU00740
MXU00750
MXU00760
MXU00770
MXU00780
MXU00790
MXU00800
MXU00810
MXU00820
MXU00830
MXU00840
MXU00850
MXU00860
MXU00870
MXU00880
MXU00890
MXU00900
MXU00910
MXU00920
MXU00930
MXU00940

```


C	AUNFRM:	FORMAT TO BE USED TO READ THE ANGLE UNIT TO OR WRITE IT FROM	MXU00950
C		A SEQUENTIAL FILE	MXU00960
C	DEFAUN:	THE DEFAULT VALUE OF THE ANGLE UNIT	MXU00970
C	TPUNTS INITIALIZED IN BLOCK DATA	MXU00980
C	PSIPUN=	1 THE PROGRAM STANDARD TEMPERATURE UNIT IS CELCIUS DEGREES	MXU00990
C	UIOIPU:	DENOTES THE INPUT/OUTPUT TEMPERATURE UNIT	MXU01000
C	= 1	IF THE INPUT/OUTPUT TEMPERATURE UNIT IS CELCIUS DEGREES	MXU01010
C	= 2	FAHRENHEIT DEGREES	MXU01020
C	= 3	KELVIN DEGREES	MXU01030
C	= 4	RANKINE DEGREES	MXU01040
C	IPINIO INITIALIZED IN BLOCK DATA	MXU01050
C	DBIPUN:	WHERE THE DATABASE NAME FOR THE TEMPERATURE UNIT IS STORED	MXU01060
C	DBIPUC:	WHERE INFORMATION FOR IDENTIFYING THE TEMPERATURE UNIT	MXU01070
C		VARIABLE IS STORED	MXU01080
C	TPUFRM:	FORMAT TO BE USED TO READ THE TEMPERATURE UNIT FROM OR WRITE	MXU01090
C		IT TO A SEQUENTIAL FILE	MXU01100
C	DEFTPU:	THE DEFAULT VALUE OF THE TEMPERATURE UNIT	MXU01110
C	-----	SUBPROGRAMS AND FUNCTIONS CALLED-----	MXU01120
C	DLX		MXU01130
C	SIRPAK		MXU01140
C	LHOVIC		MXU01150
C	MINUIN		MXU01160
C	DEX LIBRARY		MXU01170
C	NONE		MXU01180
C	MODUIT		MXU01190
C	LUNII		MXU01200
C	TUNII		MXU01210
C	FUNII		MXU01220
C	AUNII		MXU01230
C	TPUNII		MXU01240
C	-----		MXU01250
C			MXU01260
C			MXU01270
C			MXU01280
C			MXU01290
C			MXU01300
C			MXU01310
C			MXU01320
C			MXU01330
C			MXU01340
C			MXU01350
C			MXU01360
C			MXU01370
C			MXU01380
C			MXU01390
C			MXU01400
C			MXU01410

C	LABELLD	COMMONS
C		COMMON /DIALGI/ MIERSE
C		COMMON /INOUT/ IMODE, OMODE
C		COMMON /MDNGPW/ NCPW
C		COMMON /REFNOS/ RNRFIL, RNWFIL
C		COMMON /LUNITS/ PSILUN, UIOLUN
C		COMMON /TUNITS/ PSTTUN, UIOTUN
C		COMMON /FUNITS/ PSTFUN, UIOFUN
C		COMMON /AUNITS/ PSTAUN, UIOAUN
C		COMMON /TPUNTS/ PSTPUN, UIOTPU
C		COMMON /LUINFO/ DBLUNN, DBLUNC, LUNFRM, DEFLUN
C		COMMON /TUINFO/ DBTUNN, DBTUNC, TUNFRM, DEFTUN
C		COMMON /FUINFO/ DBFUNN, DBFUNC, FUNFRM, DEEFUN
C		COMMON /AUINFO/ DBAUNN, DBAUNC, AUNFRM, DEFAUN


```

COMMON /IPINFO/ DBTPUN, DBTPUC, TPUFRM, DEFTPU
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
      INTEGER IOILAG
      INTEGER IMODE, OMODE, IOMODE
      INTEGER MESS(11), IMS, NCPW
      INTEGER ITEM, MENUIN, MENUNM(2), NITEMS, ITEMS(14)
      INTEGER UIOLUN, UIOTUN, UIOFUN, UIOAUN, UIOTPU
      INTEGER DBLUNN(2), DBFUNN(2), DBAUNN(2), DBTPUN(2)
      INTEGER DBLUNC(16), DBTUNC(16), DBAUNC(16), DBTPUC(16)
      INTEGER IUNFRM(2), IUNFRM(2), FUNFRM(2), AUNFRM(2), TPUFRM(2)
      INTEGER DEFTUN, DEFTUN, DEFTUN, DEFAUN, DEFTPU
      INTEGER RNRFT1, RNRFT1, RNRFT1
      INTEGER PSTTUN, PSTTUN, PSTFUN, PSTAUN, PSTPUN
      INTEGER PMES(16)
      INTEGER READ(2), EDIT(2), WRITE(2)
      LOGICAL MTRSE, PMPREP
      LOGICAL CALALL, LOCALL
      LOGICAL LOGVAL, IMOVEC
C
C VARIABLE DATA DEFINITIONS
C
      DATA IMS /11/
      DATA MINUMM/4HUNIT,4H /
      DATA NITEMS/11/
      DATA ITEMS /4HALL ,4H ,
      2 4HLENG,4HTH ,
      3 4HTIME,4H ,
      4 4HFORC,4HE ,
      5 4HPLAN,4HEANG,
      6 4HTEMP,4H ,
      7 4HDONE,4H /
      DATA READ /4HREAD,4H.< /
      DATA EDIT /4HEDIT,4H.< /
      DATA WRITE/4HWRTT,4HE.< /
C
C INITIALIZE THE VALUE OF PMPREP.
C
      PMPREP=.TRUE.
C
C SET THE VALUE OF IOMODE AND RNFILE ACCORDING TO WHETHER THE
C USER WISHES TO READ, EDIT OR WRITE.
C
      IF (IOFLAG.EQ.3) GO TO 10
      IOMODE=IMODE
      RNFILE=RNRFIL

```

```

MXU01420
MXU01430
MXU01440
MXU01450
MXU01460
MXU01470
MXU01480
MXU01490
MXU01500
MXU01510
MXU01520
MXU01530
MXU01540
MXU01550
MXU01560
MXU01570
MXU01580
MXU01590
MXU01600
MXU01610
MXU01620
MXU01630
MXU01640
MXU01650
MXU01660
MXU01670
MXU01680
MXU01690
MXU01700
MXU01710
MXU01720
MXU01730
MXU01740
MXU01750
MXU01760
MXU01770
MXU01780
MXU01790
MXU01800
MXU01810
MXU01820
MXU01830
MXU01840
MXU01850
MXU01860
MXU01870
MXU01880

```



```

4      RNFILE, LUNFRM,
5      DEFUN)
      IF (LOCAL) GO TO 300
      IF (.NOT. CALALL) GO TO 20
      CALALL=.FALSE.
      GO TO 700

C
C READ, EDIT OR WRITE THE TIME UNIT.
C
      300 CONTINUE
      CALL FUNT(UIOIUN, LOCAL,
1          IOFLAG, IOMODE, MTERSE, NCPW,
2          DBTUNN, DBFUNC,
3          PMPREP, PMES,
4          RNFILE, LUNFRM,
5          DEFUN)
      IF (LOCAL) GO TO 400
      IF (.NOT. CALALL) GO TO 20
      CALALL=.FALSE.
      GO TO 700

C
C READ, EDIT OR WRITE THE FORCE UNIT.
C
      400 CONTINUE
      CALL FUNT(UIOIUN, LOCAL,
1          IOFLAG, IOMODE, MTERSE, NCPW,
2          DBFUNN, DBFUNC,
3          PMPREP, PMES,
4          RNFILE, LUNFRM,
5          DEFFUN)
      IF (LOCAL) GO TO 500
      IF (.NOT. CALALL) GO TO 20
      CALALL=.FALSE.
      GO TO 700

C
C READ, EDIT OR WRITE THE PLANE ANGLE UNIT.
C
      500 CONTINUE
      CALL AUNIT(UIOAUN, LOCAL,
1          IOFLAG, IOMODE, MTERSE, NCPW,
2          DBAUNN, DBAUNC,
3          PMPREP, PMES,
4          RNFILE, AUNFRM,
5          DEFAUN)
      IF (LOCAL) GO TO 600
      IF (.NOT. CALALL) GO TO 20
      CALALL=.FALSE.

```

```

MXU02360
MXU02370
MXU02380
MXU02390
MXU02400
MXU02410
MXU02420
MXU02430
MXU02440
MXU02450
MXU02460
MXU02470
MXU02480
MXU02490
MXU02500
MXU02510
MXU02520
MXU02530
MXU02540
MXU02550
MXU02560
MXU02570
MXU02580
MXU02590
MXU02600
MXU02610
MXU02620
MXU02630
MXU02640
MXU02650
MXU02660
MXU02670
MXU02680
MXU02690
MXU02700
MXU02710
MXU02720
MXU02730
MXU02740
MXU02750
MXU02760
MXU02770
MXU02780
MXU02790
MXU02800
MXU02810
MXU02820

```



```

MXU02830
MXU02840
MXU02850
MXU02860
MXU02870
MXU02880
MXU02890
MXU02900
MXU02910
MXU02920
MXU02930
MXU02940
MXU02950
MXU02960
MXU02970
MXU02980
MXU02990
MXU03000
MXU03010
MXU03020

```

```

      GO TO 700
C
C READ, EDIT OR WRITE THE TEMPERATURE UNIT.
C
600 CONTINUE
    CALL TPUNIT(UOTPU,LOCAL,
     1 IOFLAG,IOMODE,MTERSE,NCPW,
     2 DBTPUN,DBTPUC,
     3 PMPREP,PMES,
     4 RNFILE,PUFRM,
     5 DEIPU)
    IF (LOCAL) GO TO 700
    IF (.NOT.CALALL) GO TO 20
    CALALL=.FALSE.
C
C RETURN CONTROL TO THE CALLING PROGRAM.
C
700 CONTINUE
    RETURN
    END

```



```

C-----CUBE MODULE SUBPROGRAM-----
SUBROUTINE DIMENS(ALLFLG,IOFLAG)
C-----SUBPROGRAM DESCRIPTION-----
C SUBROUTINE DIMENS FIRST CALLS UNIT1F TO OBTAIN THE MULTIPLI-
C CATIVE LENGTH CONVERSION FACTOR AND THE NAMES OF THE LENGTH UNITS
C TO BE USED DURING INPUT. DIMENS THEN PROVIDES A MENU FROM
C FROM WHICH THE USER SELECTS WHICH DIMENSION IT IS DESIRED TO READ,
C EDIT OR WRITE. THE CHOICES ARE:
C ALL DIMENSIONS
C LENGTH
C WIDTH
C HEIGHT
C IF THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE, THE LOCAL
C ALL OPTION IS SET TO .TRUE. UPON INVOKING THIS SUBROUTINE AND THE
C MENU IS NOT DISPLAYED.
C-----SUBPROGRAM ASSUMPTIONS-----
C NONE YET
C-----OUTPUT VARIABLES-----
C ALLFLG: .TRUE. IF THE INPUT VALUE OF ALLFLG WAS .TRUE. AND NO
C ERROR OCCURED WHEN READING A UNIT OR EDITING ANY
C DIMENSION
C .FALSE. IF THE INPUT VALUE OF ALLFLG WAS .FALSE. OR AN
C ERROR OCCURED WHEN READING A UNIT OR EDITING A
C DIMENSION
C-----INPUT VARIABLES-----
C ALLFLG: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C : .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE
C IOFLAG=1 IF THE USER WISHES TO INVOKE RSCDDR
C .2 RSCDDT
C .3 RSCDMP
C-----LABELED COMMON VARIABLES-----
C LABELD COMMONS DIALGF,INOUTF,MDCNPF AND REFNOS HAVE BEEN DEFINED
C IN SUBROUTINE MAINPG. LABELED COMMON UNITS HAS BEEN DEFINED IN
C SUBROUTINE MXUNIT.
C.....UNITO INITIALIZED IN BLOCK DATA
C L : LENGTH OF THE CUBE IN PROGRAM STANDARD UNITS
C LENNAM: DATABASE NAME FOR LENGTH
C LMORCN: DATABASE COMMENT DESCRIBING LENGTH
C DEFALT: THE DEFAULT VALUE FOR LENGTH
C.....VINIO INITIALIZED IN BLOCK DATA
C W : WIDTH OF THE CUBE IN PROGRAM STANDARD UNITS
C WEINAM: DATABASE NAME FOR WIDTH
C WMORCN: DATABASE COMMENT DESCRIBING WIDTH
C DEFALT: THE DEFAULT VALUE FOR WIDTH
C.....HIINTO INITIALIZED IN BLOCK DATA
C H : HEIGHT OF THE CUBE IN PROGRAM STANDARD UNITS
C HEINAM: DATABASE NAME FOR HEIGHT

```

```

DIM00010
DIM00020
DIM00030
DIM00040
DIM00050
DIM00060
DIM00070
DIM00080
DIM00090
DIM00100
DIM00110
DIM00120
DIM00130
DIM00140
DIM00150
DIM00160
DIM00170
DIM00180
DIM00190
DIM00200
DIM00210
DIM00220
DIM00230
DIM00240
DIM00250
DIM00260
DIM00270
DIM00280
DIM00290
DIM00300
DIM00310
DIM00320
DIM00330
DIM00340
DIM00350
DIM00360
DIM00370
DIM00380
DIM00390
DIM00400
DIM00410
DIM00420
DIM00430
DIM00440
DIM00450
DIM00460
DIM00470

```



```

C IMORGN: DATABASE COMMENT DESCRIBING HEIGHT
C NDETH : THE NUMBER OF HEIGHT DEFAULT VALUES
C DEFALH: THE DETAIL VALUES FOR HEIGHT
C.....DIMIRM INITIALIZED IN BLOCK DATA
C LWRM : FORMAT TO BE USED FOR THE LENGTH AND WIDTH DIMENSIONS WHEN
C         READING FROM OR WRITING TO A SEQUENTIAL FILE
C IIRM : FORMAT TO BE USED FOR THE HEIGHT DIMENSION WHEN READING FROM
C         OR WRITING TO A SEQUENTIAL FILE
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----
C DEX
C     SIRPAK
C     MISOUT
C     MINUIN
C     DEX LIBRARY
C     UNITIF
C     RATLDR
C     RARDET
C     RARDEP
C     RSCDDR
C     RSCDDT
C     RSCDDP
C MODULI
C     NONE
C-----
C LABELD COMMONS
C
C     COMMON /DIALGI/ MTRSE
C     COMMON /INOUTI/ IMODE,OMODE
C     COMMON /MDNCPW/ NCPW
C     COMMON /RETROS/ RNRFIL,RNWFIL
C     COMMON /LUNITS/ PSTLUN,UIOLUN
C     COMMON /LINFO/ L,LENNAM,LMORGN,DEFAL
C     COMMON /WINFO/ W,WIDNAM,WMORGN,DEFALW
C     COMMON /HINFO/ H(4),HLEINAM,IMORCN,NDETH,DETH(4)
C     COMMON /DIMFRM/ LWFRM,IIFRM
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
C     INTEGER IOLLAG, IMODE,OMODE,NCPW
C     INTEGER RNRFIL,RNWFIL
C     INTEGER MXIOCT,NFROM,NCOT,EFROM,NT0
C     INTEGER LWFRM(2),IIFRM(3)
C     INTEGER NAMLO2(1),NAMLO6(2),NAML12(3)
C     INTEGER PSTLUN,UIOLUN
C     INTEGER MESS(14),LMS
C     INTEGER LENNAM(2),WIDNAM(2),HLEINAM(2)
C     INTEGER PHIS(16),LMORGN(16),WMORGN(16),IMORCN(16)
C     INTEGER MENUDM(2),NITEMS,ITEMS(10),ITEM
C     INTEGER READ(2),EDIT(2),WRITE(2)

```



```

LOGICAL MIERSE,VITAL,PMREP
LOGICAL LOGVAL,MOVEC
LOGICAL ALLFLG,LOCAL
LOGICAL UNITLF
LOGICAL RSCIDR,RSCEDT,RSCDMP,RATLDR,RAREDIT,RARDMP
REAL L,W,H,DEFTL,DEFALW,DEFALH
REAL CONVLM,CONVLA

C
C VARIABLE DATA DEFINITIONS
C
DATA MENUDM/4HIDIME,4HINSIO/
DATA NITEMS/5/
DATA ITEMS/4HALL,4H1
,4HLENG,4HETH
,
1
2
3
4
4HVIDT,4HHH
,
4HIFLG,4HHIT
,
4HDONE,4H /
DATA CONVLA/0.0/
DATA LMS/14/
DATA READ /4HREAD,4H1,4 /
DATA EDIT /4HEDIT,4H1,4 /
DATA WRITE/4HWRITE,4H1,4 /
DATA MXIOGT,NFROM,NGOT /4,1,4/

C
C DETERMINE THE NAMLS OF THE UNITS FOR THE INPUT DIMENSIONS AND THE
C MULTIPLICATIVE CONVERSION FACTOR TO CONVERT THE INPUT UNITS TO THE
C PROGRAM STANDARD UNITS.
C
LOGVAL=UNITLF(CONVLM,NAML02,NAML06,NAML12,ALLFLG,
1 PSTLUN,UIOLUN,NCPW)
IF (.NOT.LOGVAL) GO TO 99999

C
C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF THE LOCAL ALL OPTION IS SET IN THIS MANNER, MENU
C 'DIMENSION' IS NOT DEFINED BY THE INVOCATION OF THIS SUBROUTINE.
C
LOCAL=ALLFLG
IF (LOCAL) GO TO 200

C
C PREPARE A PROMPTING MESSAGE FOR MENU 'DIMENSIO'.
C
5 CONTINUE
IF (MIERSE) GO TO 40
CALL STRPAK(MESS,LMS,4H1,4H2,33HSELECT THE DESIRED DIMENSION TO< ) DIM01380
GO TO (10,20,30),10FLAG DIM01390
10 CONTINUE DIM01400
LOGVAL=LMOVEC(READ,1,6,NCPW,MESS,33,NCPW) DIM01410

```

```

DIM00950
DIM00960
DIM00970
DIM00980
DIM00990
DIM01000
DIM01010
DIM01020
DIM01030
DIM01040
DIM01050
DIM01060
DIM01070
DIM01080
DIM01090
DIM01100
DIM01110
DIM01120
DIM01130
DIM01140
DIM01150
DIM01160
DIM01170
DIM01180
DIM01190
DIM01200
DIM01210
DIM01220
DIM01230
DIM01240
DIM01250
DIM01260
DIM01270
DIM01280
DIM01290
DIM01300
DIM01310
DIM01320
DIM01330
DIM01340
DIM01350
DIM01360
DIM01370
DIM01380
DIM01390
DIM01400
DIM01410

```



```

GO TO 50
20 CONTINUE
  LOGVAL=LMOVEC(EDIT,1,6,NCPW,MESS,33,NCPW)
GO TO 50
30 CONTINUE
  LOGVAL=LMOVEC(WRITE,1,7,NCPW,MESS,33,NCPW)
GO TO 50
40 CONTINUE
  CALL STRPAK(MESS,LMS,4HK ,17WHICH DIMENSION?4)
50 CONTINUE
  ITEM=MENUIN(MENUDM,NITEMS,ITEMS,MESS)
  GO TO (100,200,300,400,99999),ITEM

C
C SET LOCAL ALL OPTION.
C
100 CONTINUE
  LOCAL=.TRUE.

C
C DETERMINI WHICH OPERATION TO PERFORM ON THE LENGTH DIMENSION
C AND BRANCH ACCORDINGLY.
C
200 CONTINUE
  GO TO (210,220,230),IOFLAG

C
C READ INPUT LENGTH.
C
210 CONTINUE
  LOGVAL=RSCIDR(L,LOCAL,
1  MIERSE,IMODE,NCPW,
2  LENNAM,CONVLM,CONVLA,NAML12,.FALSE.,
3  .TRUE.,PMES,LMORGN,
4  RNRFIL,LWFRM,DEFAULT)
  IF (LOCAL) GO TO 300
  IF (.NOT.ALFLG) GO TO 5
  ALFLG=.FALSE.
  GO TO 99999

C
C EDIT LENGTH.
C
220 CONTINUE
  LOGVAL=RSCGDT(L,LOCAL,
1  MIERSE,NCPW,
2  LENNAM,CONVLM,CONVLA,NAML12,
3  .TRUE.,PMES,LMORGN,
4  RNRFIL,LWFRM,
5  DEFAULT)
  IF (LOCAL) GO TO 300

```

```

DIM01420
DIM01430
DIM01440
DIM01450
DIM01460
DIM01470
DIM01480
DIM01490
DIM01500
DIM01510
DIM01520
DIM01530
DIM01540
DIM01550
DIM01560
DIM01570
DIM01580
DIM01590
DIM01600
DIM01610
DIM01620
DIM01630
DIM01640
DIM01650
DIM01660
DIM01670
DIM01680
DIM01690
DIM01700
DIM01710
DIM01720
DIM01730
DIM01740
DIM01750
DIM01760
DIM01770
DIM01780
DIM01790
DIM01800
DIM01810
DIM01820
DIM01830
DIM01840
DIM01850
DIM01860
DIM01870
DIM01880

```



```

      IF (.NOT.ALFLG) GO TO 5
      ALLFG=.FALSE.
      GO TO 99999
C
C   WRITE LENGTH.
C
230  CONTINUE
      LOGVAL=RSCDMP(LOCALL,
1      MTERSE, OMODE, NCPW,
2      L, LFNNAM, CONVLM, CONVLA, NAML12,
3      .TRUE., PMES, LMORGN,
4      RNRFIL, LWFRM)
      IF (LOCALL) GO TO 300
      IF (.NOT.ALFLG) GO TO 5
      ALLFG=.FALSE.
      GO TO 99999
C
C   DETERMINE WHICH OPERATION TO PERFORM ON THE WIDTH DIMENSION AND
C   BRANCH ACCORDINGLY.
C
300  CONTINUE
      GO TO (310, 320, 330), IOFLAG
C
C   READ WIDTH.
C
310  CONTINUE
      LOGVAL=RSCDDR(W, LOCALL,
1      MTERSE, IMODF, NCPW,
2      WIDNAM, CONVLM, CONVLA, NAML12, .FALSE.,
3      .TRUE., PMES, WMORGN,
4      RNRFIL, LWFRM, DEFALW)
      IF (LOCALL) GO TO 400
      IF (.NOT.ALFLG) GO TO 5
      ALLFG=.FALSE.
      GO TO 99999
C
C   EDIT WIDTH.
C
320  CONTINUE
      LOGVAL=RSCEDI(W, LOCALL,
1      MTERSE, NCPW,
2      WIDNAM, CONVLM, CONVLA, NAML12,
3      .TRUE., PMES, WMORGN,
4      RNRFIL, LWFRM,
5      DEFALW)
      IF (LOCALL) GO TO 400
      IF (.NOT.ALFLG) GO TO 5

```

```

DIM01890
DIM01900
DIM01910
DIM01920
DIM01930
DIM01940
DIM01950
DIM01960
DIM01970
DIM01980
DIM01990
DIM02000
DIM02010
DIM02020
DIM02030
DIM02040
DIM02050
DIM02060
DIM02070
DIM02080
DIM02090
DIM02100
DIM02110
DIM02120
DIM02130
DIM02140
DIM02150
DIM02160
DIM02170
DIM02180
DIM02190
DIM02200
DIM02210
DIM02220
DIM02230
DIM02240
DIM02250
DIM02260
DIM02270
DIM02280
DIM02290
DIM02300
DIM02310
DIM02320
DIM02330
DIM02340
DIM02350

```



```

C      ALLFLG=.FALSE.
C      GO TO 99999
C
C      WRITE WIDTHH.
C
C      330 CONTINUE
C      LOGVAL=RSCDMP(LOCALL,
C      1      MIERSE,OMODF,NCIPW,
C      2      W,WIDNAM,CONVLM,CONVLA,NAML12,
C      3      .TRUE.,PMES,IMOIRGN,
C      4      RNRFIL,LWFRM)
C      IF (LOCALL) GO TO 400
C      IF (.NOT.ALLFLG) GO TO 5
C      ALLFLG=.FALSE.
C      GO TO 99999
C
C      DETERMINE WHICH OPERATION TO PERFORM ON THE HEIGHT DIMENSION
C      AND BRANCH ACCORDINGLY.
C
C      400 CONTINUE
C      GO TO (410,420,430), IOFLAG
C
C      READ HEIGHT.
C
C      410 CONTINUE
C      LOGVAL=RATLDR(II,LOCALL,NGOI,
C      1      MIERSE,IMODE,NCIPW,
C      2      HEINAM,MXIOGT,CONVLM,CONVLA,NAML12,.FALSE.,
C      3      .TRUE.,PMES,IMOIRGN,
C      4      RNRFIL,IIFRM,NDEFH,DEFALH)
C      IF (LOCALL) GO TO 99999
C      IF (.NOT.ALLFLG) GO TO 5
C      ALLFLG=.FALSE.
C      GO TO 99999
C
C      EDIT HEIGHT.
C
C      420 CONTINUE
C      LOGVAL=RARED(II,LOCALL,
C      1      MIERSE,NCIPW,
C      2      HEINAM,EFROM,NIO,
C      3      CONVLM,CONVLA,NAML12,
C      4      .TRUE.,PMES,IMOIRGN,
C      5      RNRFIL,IIFRM,
C      6      NDEFH,DEFALH)
C      IF (LOCALL) GO TO 99999
C      IF (.NOT.ALLFLG) GO TO 5

```

```

DIM02360
DIM02370
DIM02380
DIM02390
DIM02400
DIM02410
DIM02420
DIM02430
DIM02440
DIM02450
DIM02460
DIM02470
DIM02480
DIM02490
DIM02500
DIM02510
DIM02520
DIM02530
DIM02540
DIM02550
DIM02560
DIM02570
DIM02580
DIM02590
DIM02600
DIM02610
DIM02620
DIM02630
DIM02640
DIM02650
DIM02660
DIM02670
DIM02680
DIM02690
DIM02700
DIM02710
DIM02720
DIM02730
DIM02740
DIM02750
DIM02760
DIM02770
DIM02780
DIM02790
DIM02800
DIM02810
DIM02820

```


DIM02830
 DIM02840
 DIM02850
 DIM02860
 DIM02870
 DIM02880
 DIM02890
 DIM02900
 DIM02910
 DIM02920
 DIM02930
 DIM02940
 DIM02950
 DIM02960
 DIM02970
 DIM02980
 DIM02990
 DIM03000
 DIM03010
 DIM03020
 DIM03030

```

ALLIG=.FALSE.
GO TO 9999
C WRITE HEICHI.
C
430 CONTINUE
    LOGVAL=RARDMP(LOCAL,
      1 MTERSE, OMODE, NCPW,
      2 H, HEINAM, NFROM, NGOI,
      3 CONVLM, CONVLA, NAML12,
      4 ,TRUE., PMES, IMORCN,
      5 RNWFL, IIFRM)
      11 (LOCAL) GO TO 9999
      11 (.NOT.ALFLG) GO TO 5
      ALLIG=.FALSE.
C RETURN TO CALLING PROGRAM.
C
9999 CONTINUE
RETURN
END

```



```

C-----CUBE MODLE SUBPROGRAM-----COM00010
C      SUBROUTINE COMPUTCOM00020
C-----SUBPROGRAM DESCRIPTION-----COM00030
C      THIS SUBROUTINE CALCULATES THE VOLUME AND WEIGHT OF A CUBE OF SALT COM00040
C      WATER.COM00050
C      THE STANDARD UNITS OF THIS SUBROUTINE ARE METERS AND KILOPONDS.COM00060
C-----LABELED COMMON VARIABLES-----COM00070
C      LABEL'D COMMONS LINFO,WINFO,IIINFO,VINFO AND WTINFO HAVE BEEN DEFINED COM00080
C      IN SUBROUTINES DIMENS AND VANDWI.COM00090
C-----COM00100
C      COM00110
C      COM00120
C      COM00130
C      COM00140
C      COM00150
C      COM00160
C      COM00170
C      COM00180
C      COM00190
C      COM00200
C      COM00210
C      COM00220
C      COM00230
C      COM00240
C      COM00250
C      COM00260
C      COM00270
C      COM00280
C      COM00290
C      COM00300
C      COM00310
C      COM00320
C      COM00330
C      COM00340
C      COM00350
C      COM00360
C      COM00370
C      COM00380
C      COM00390
C      COM00400
C      COM00410
C      COM00420

C      COMMON /LINFO/ L
C      COMMON /WINFO/ W
C      COMMON /IIINFO/ II(h)
C      COMMON /VINFO/ V(h)
C      COMMON /WTINFO/ WT(h)

C      VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
C      REAL L,W,II,V,WT
C      REAL RHO
C
C      VARIABLE DATA DEFINITIONS
C
C      DATA RHO/1023.2/
C
C      CALCULATE THE VOLUME IN CUBIC METERS.
C
C      DO 10 I=1,h
C      V(I) = L*W*H(I)
C      10 CONTINUE
C
C      CALCULATE THE WEIGHT OF THE CUBE IF THE WEIGHT DENSITY OF SALT
C      WATER IS 1023.2 KILOPONDS/M**3.
C
C      DO 20 I=1,h
C      WT(I) = V(I)*RHO
C      20 CONTINUE
C      RETURN
C      END

```



```

C-----CUBE MODULE SUBPROGRAM-----
C SUBROUTINE OUTPUT(CALL, IOFLAG)
C-----SUBPROGRAM DESCRIPTION-----
C SUBROUTINE OUTPUT PROVIDES THE USER WITH A MENU FROM WHICH TO
C CHOOSE WHICH MODULE SEGMENT IT IS DESIRED TO OPERATE NEXT. THE
C CHOICES ARE:
C ALL MODULE OUTPUT VARIABLES
C THE MODULE UNITS TO BE USED DURING INPUT AND OUTPUT
C THE MODULE RESULTS
C THE UNITS MODULE ALLOWS THE USER TO SPECIFY THE LENGTH AND FORCE
C UNITS TO BE USED DURING INPUT AND OUTPUT. THE RESULTS MODULE
C ALLOWS THE USER TO READ, EDIT OR WRITE THE CUBE VOLUME AND WEIGHT.
C IF THE CALLING PROGRAM ALL OPTION IS ACTIVE, THE LOCAL ALL
C OPTION IS SET TO .TRUE. UPON INVOKING THIS SUBROUTINE. IN THIS CASE
C MENU 'OUTPUT' IS NOT DEFINED.
C-----SUBPROGRAM ASSUMPTIONS-----
C THE UNITS MODULE SHOULD BE ACCESSED IF EITHER UTOLOW OR UTOFUP,
C WHICH ARE USED TO FIND THE VOLUME AND WEIGHT UNITS RESPECTIVELY, ARE
C TO BE DIFFERENT FROM THAT WHICH WAS ESTABLISHED BY EITHER THE BLOCK
C DATA SUBPROGRAM OR DURING THE INPUT SEQUENCE.
C-----OUTPUT VARIABLES-----
C CALL: .TRUE. IF THE INPUT VALUE OF CALL WAS .TRUE. AND NO ERROR
C .FALSE. IF THE INPUT VALUE OF CALL WAS .FALSE. OR AN ERROR
C OCCURRED WHEN READING OR EDITING AN OUTPUT VARIABLE
C-----INPUT VARIABLES-----
C CALL: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE
C IOFLAG=1 IF THE USER WISHES TO READ THE VARIABLES
C 2 EDIT
C 3 WRITE
C-----LABELLED COMMON VARIABLES-----
C LABELLED COMMON DIALCF AND MDNCPW HAVE BEEN DEFINED IN SUBROUTINE
C MAINPG.
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----
C DEX STRPAK
C LMOVIC
C MENUIN
C DEX LIBRARY
C NONE
C MODULE
C MXUNIT
C VANDWT
C-----
C Labeled Commons
C COMMON /DIALCF/ MTERSE

```



```

COMMON /HIDNCPW/ NCPW
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
C
C   INTEGER IOFLAG
C   INTEGER MENU(2), NITEMS, ITEMS(8), ITEM
C   INTEGER MESS(16), LMS, NCPW
C   INTEGER READ(2), EDIT(2), WRITE(2)
C   LOGICAL CALALL, LOCAL
C   LOGICAL MTRSE, LOGVAL, LMOVEC
C
C VARIABLE DATA DEFINITIONS
C
C   DATA LMS/16/
C   DATA MENU/MHOUTP, MHUT /
C   DATA NITEMS/4/
C   DATA ITEMS/MHALL, MH ,
C   1      MHUNT, MHS ,
C   2      MIRSU, MRLTS ;
C   3      MHDONE, MH /
C   DATA LOCAL/.FALSE./
C   DATA READ /MHREAD, MH.< /
C   DATA EDIT /MHEDIT, MH.< /
C   DATA WRITE/MHWRTT, MHE.< /
C
C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF THE LOCAL ALL OPTION IS SET IN THIS MANNER, MENU
C 'OUTPUT' WILL NOT BE DEFINED BY THE INVOCATION OF THIS SUBROUTINE.
C
C   LOCAL=CALL
C   IF (LOCAL) GO TO 200
C
C PREPARE A PROMPTING MESSAGE FOR MENU 'OUTPUT' AND THEN PROVIDE THE
C MENU TO THE USER.
C
C   5 CONTINUE
C   CALL STRPAK(MESS, LMS, MHK , MHSELECT WHICH OUTPUT VARIABLE SEGMENT
C   IF TO<)
C   GO TO (10, 20, 30), IOFLAG
C   10 LOGVAL=LMOVEC(READ, 1, 6, NCPW, MESS, 41, NCPW)
C   GO TO 50
C   20 LOGVAL=LMOVEC(EDIT, 1, 6, NCPW, MESS, 41, NCPW)
C   GO TO 50
C   30 LOGVAL=LMOVEC(WRITE, 1, 7, NCPW, MESS, 41, NCPW)
C   50 CONTINUE
C   ITEM=MENU(MENU, NITEMS, ITEMS, MESS)
C   GO TO (100, 200, 300, 400), ITEM
C
C SET THE OUTPUT ALL OPTION.

```

```

OUT00500
OUT00510
OUT00520
OUT00530
OUT00540
OUT00550
OUT00560
OUT00570
OUT00580
OUT00590
OUT00600
OUT00610
OUT00620
OUT00630
OUT00640
OUT00650
OUT00660
OUT00670
OUT00680
OUT00690
OUT00700
OUT00710
OUT00720
OUT00730
OUT00740
OUT00750
OUT00760
OUT00770
OUT00780
OUT00790
OUT00800
OUT00810
OUT00820
OUT00830
OUT00840
OUT00850
OUT00860
OUT00870
OUT00880
OUT00890
OUT00900
OUT00910
OUT00920
OUT00930
OUT00940
OUT00950
OUT00960
OUT00970
OUT00980

```



```

C      100 CONTINUE
C      LOCAL=.TRUE.
C
C      C READ, EDIT OR WRITE THE INPUT/OUTPUT MODULE UNITS.
C
C      200 CONTINUE
C      CALL MXUNIT(LOCAL,IOFLAG)
C      IF (LOCAL) GO TO 300
C      IF (.NOT.CALALL) GO TO 50
C      CALALL=.FALSE.
C      GO TO 400
C
C      C CALL VANDWT TO READ, EDIT OR WRITE THE CUBE VOLUME AND WEIGHT.
C
C      300 CONTINUE
C      CALL VANDWT(LOCAL,IOFLAG)
C      IF (LOCAL) GO TO 400
C      IF (.NOT.CALALL) GO TO 50
C      CALALL=.FALSE.
C
C      C RETURN CONTROL TO CALLING PROGRAM.
C
C      400 CONTINUE
C      RETURN
C      END

```

OUT00990
 OUT01000
 OUT01010
 OUT01020
 OUT01030
 OUT01040
 OUT01050
 OUT01060
 OUT01070
 OUT01080
 OUT01090
 OUT01100
 OUT01110
 OUT01120
 OUT01130
 OUT01140
 OUT01150
 OUT01160
 OUT01170
 OUT01180
 OUT01190
 OUT01200
 OUT01210
 OUT01220
 OUT01230
 OUT01240


```

C-----CUBE MODULE SUBPROGRAM-----VAN00010
C      SUBROUTINE VANDWT(ALLFLG,IOTLAC)VAN00020
C-----SUBPROGRAM DESCRIPTION-----VAN00030
C      SUBROUTINE VANDWT FIRST CALLS UNITL, UNITFF AND UVOL TO OBTAIN THEVAN00040
C      MULTIPLICATIVE LENGTH, FORCE (WEIGHT) AND VOLUME CONVERSION FACTORSVAN00050
C      AND THE NAMES OF THE FORCE AND VOLUME UNITS TO BE USED DURING OUTPUT.VAN00060
C      VANDWT THEN PROVIDES A MENU FROM WHICH THE USER SELECTS WHICHVAN00070
C      PROGRAM COMPUTATION RESULTS IT IS DESIRED TO READ, EDIT OR WRITE.VAN00080
C      THE CHOICES ARE:VAN00090
C      ALL RESULTSVAN00100
C      VOLUMEVAN00110
C      WEIGHTVAN00120
C      VANDWT THEN CALLS RATLDR, RAREDT, OR RARDMP AS NECESSARY FOR EACHVAN00130
C      VARIABLE.VAN00140
C      IF THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE, THE LOCAL ALLVAN00150
C      OPTION IS SET TO .TRUE. IMMEDIATELY AFTER THE UNITS INFORMATION ISVAN00160
C      OBTAINED. IN THIS CASE, MENU 'RESULTS' IS NOT INVOKED.VAN00170
C-----SUBPROGRAM ASSUMPTIONS-----VAN00180
C      NONE YETVAN00190
C-----OUTPUT VARIABLES-----VAN00200
C      ALLFLG: .TRUE. IF THE INPUT VALUE OF ALLFLG WAS .TRUE. AND NO ERRORVAN00210
C      OCCURED WHEN READING ANY UNIT OR EDITING ANY PROGRAMVAN00220
C      RESULTVAN00230
C      .FALSE. IF THE INPUT VALUE OF ALLFLG WAS .FALSE. OR AN ERRORVAN00240
C      OCCURED WHEN READING A UNIT OR EDITING A PROGRAMVAN00250
C      RESULTVAN00260
C-----INPUT VARIABLES-----VAN00270
C      ALLFLG: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVEVAN00280
C      : .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVEVAN00290
C      IOTLAC=1 IF THE USER WISHES TO INVOKE RAREDTVAN00300
C      2VAN00310
C      3RARDMPVAN00320
C-----LABELED COMMON VARIABLES-----VAN00330
C      LABEL'D COMMONS DIALGF, INOUTT, MDNCPW AND REFNOS HAVE BEEN DEFINEDVAN00340
C      IN SUBROUTINE MAINPG. LABELED COMMONS LUNITS AND FUNITS HAVE BEENVAN00350
C      DEFINED IN SUBROUTINE MXUNIT.VAN00360
C      .....VINFO INITIALIZED IN BLOCK DATAVAN00370
C      C V : VOLUME OF THE CUBE IN PROGRAM STANDARD UNITSVAN00380
C      C VOLNAH: DATABASE NAME FOR VOLUMEVAN00390
C      C VMORCN: COMMENT DESCRIBING VOLUMEVAN00400
C      C NDEFV : THE NUMBER OF VOLUME DEFAULT VALUESVAN00410
C      C DEFALV: THE DEFAULT VALUES FOR VOLUMEVAN00420
C      C .....WINFO INITIALIZED IN BLOCK DATAVAN00430
C      C WT : WEIGHT OF THE CUBE IN PROGRAM STANDARD UNITSVAN00440
C      C WEINAH: DATABASE NAME FOR WEIGHTVAN00450
C      C WTMRCN: COMMENT DESCRIBING WEIGHTVAN00460
C      C NDEFWT: THE NUMBER OF WEIGHT DEFAULT VALUESVAN00470
C      C DFALWT: THE DEFAULT VALUES FOR WEIGHTVAN00480
C      C .....ANSIRM INITIALIZED IN BLOCK DATAVAN00490

```



```

C ANSRM: FORMAT TO BE USED FOR THE CUBE PROPERTIES WHEN READING FROM
C OR WRITING TO A SEQUENTIAL FILE
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----
C DEX
C SURPAK
C IMOVIC
C MI SOUT
C MINUIN
C DEX LIBRARY
C UNITLT
C UNITFF
C UVOL
C RATHDR
C RAREDT
C RARDHP
C MODUL1
C NONE
C-----
C LABELLED COMMONS
C
COMMON /DIALCF/ MIERSE
COMMON /INOUTF/ IMODE, OMODE
COMMON /MDCPWF/ NCPW
COMMON /REFNOS/ RNRTIL, RNWFIL
COMMON /LUNITS/ PSTLUN, UIOFUN
COMMON /LUNITS/ PSIFUN, UIOFUN
COMMON /VINIO/ V(h), VOLNAM, VMORGN, NDLIV, DELALV(h)
COMMON /WINEO/ W1(h), W1NAM, W1MRGN, NDEFW1, DEFALV1(h)
COMMON /ANSFRM/ ANSRM
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
INTEGER IOFLAG, IMODE, OMODE, NCPW
INTEGER RNRTIL, RNWFIL
INTEGER ANSRM(3)
INTEGER NAMLO2(1), NAMLO6(2), NAML12(3)
INTEGER NAMLO2(1), NAMEO3(1), NAME12(3)
INTEGER UNVOL(3)
INTEGER PSTLUN, UIOFUN
INTEGER PSIFUN, UIOFUN
INTEGER MESS(13), IMS
INTEGER VOINAM(2), WFINAM(2)
INTEGER PMES(16), VMORGN(16), W1MRGN(16)
INTEGER NDEFV, NDEFW1
INTEGER MENURL(2), NITEMS, IITEMS(8), IITEM
INTEGER RFAD(2), ED11(2), WRITE(2)
INTEGER MXIOGI, NIROM, NIO
INTEGER VGO1, WTCOT, FROM
VAN00500
VAN00510
VAN00520
VAN00530
VAN00540
VAN00550
VAN00560
VAN00570
VAN00580
VAN00590
VAN00600
VAN00610
VAN00620
VAN00630
VAN00640
VAN00650
VAN00660
VAN00670
VAN00680
VAN00690
VAN00700
VAN00710
VAN00720
VAN00730
VAN00740
VAN00750
VAN00760
VAN00770
VAN00780
VAN00790
VAN00800
VAN00810
VAN00820
VAN00830
VAN00840
VAN00850
VAN00860
VAN00870
VAN00880
VAN00890
VAN00900
VAN00910
VAN00920
VAN00930
VAN00940
VAN00950
VAN00960
VAN00970
VAN00980

```



```

LOGICAL MIERSE,VITAL,PMPIREP
LOGICAL LOGVAL,MOVEC
LOGICAL ALLFLG,LOCAL
LOGICAL UNITLF,UNITF1,UVOL
LOGICAL RATHDR,RARED1,RARDMP
REAL V,W1,DETA,V,DFALWT
REAL CONVLM,CONVLA
REAL CONVFM,CONVEA
REAL CNVFM,CNVEFA

C VARIABLE DATA DEFINITIONS
C
DATA MNURL/HIRESU,4HLIS /
DATA NITEMS/n/
DATA ITEMS/HIAL1,4H1,
1      4INVOLU,4HMC,
2      4HWF1G,4HHT,
3      4HDONE,4H1 /
DATA CONVLA/0.0/
DATA CONVEA/0.0/
DATA CNVEFA/0.0/
DATA LMS/13/
DATA READ /HIREAD,4H1,4 /
DATA EDIT /HREDIT,4H1,4 /
DATA WRIT/HIWRIT,4H1,4 /
DATA MXTGT,VGOT,WTGOT,FROM /4,4,4,1/

C DETERMINE THE NAMES OF THE UNITS FOR THE OUTPUT RESULTS AND THE
C MULTIPLICATIVE CONVERSION FACTORS TO CONVERT THE RESULTS IN PROGRAM
C STANDARD UNITS TO OUTPUT UNITS.
C
LOGVAL=UNITLF(CONVLM,NAMF02,NAMF06,NAMF12,ALLFLG,
1      PSTLUN,UOLUN,NCPW)
IF (.NOT.LOGVAL) GO TO 99999
LOGVAL=UVOL(CNVFVM,UNVOL,ALLFLG,
1      CONVLM,NAMF06,NCPW)
IF (.NOT.LOGVAL) GO TO 99999
LOGVAL=UNITF1(CONVFM,NAMF02,NAMF03,NAMF12,ALLFLG,
1      PSTFUN,UOFUN,NCPW)
IF (.NOT.LOGVAL) GO TO 99999

C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IS THE LOCAL ALL OPTION IS SET IN THIS MANNER, MENU
C 'RESULTS' WILL NOT BE DEFINED BY THE INVOCATION OF THIS SUBROUTINE.
C
LOCAL=ALLFLG
IF (LOCAL) GO TO 200

C PREPARE A PROMPTING MESSAGE FOR MENU 'RESULTS'.

```

```

VAN00990
VAN01000
VAN01010
VAN01020
VAN01030
VAN01040
VAN01050
VAN01060
VAN01070
VAN01080
VAN01090
VAN01100
VAN01110
VAN01120
VAN01130
VAN01140
VAN01150
VAN01160
VAN01170
VAN01180
VAN01190
VAN01200
VAN01210
VAN01220
VAN01230
VAN01240
VAN01250
VAN01260
VAN01270
VAN01280
VAN01290
VAN01300
VAN01310
VAN01320
VAN01330
VAN01340
VAN01350
VAN01360
VAN01370
VAN01380
VAN01390
VAN01400
VAN01410
VAN01420
VAN01430
VAN01440
VAN01450
VAN01460
VAN01470

```



```

C
5  CONTINUE
  IF (MTERSE) GO TO 40
  CALL SIRPAK(MESS,LMS,4HK ,30HSELECT THE DESIRED RESULT TO<)
  GO TO (10,20,30),IOFLAG
10  CONTINUE
  LOGVAL=LMOVEC(READ,1,6,NCPW,MESS,30,NCPW)
  GO TO 50
20  CONTINUE
  LOGVAL=LMOVEC(EDIT,1,6,NCPW,MESS,30,NCPW)
  GO TO 50
30  CONTINUE
  LOGVAL=LMOVEC(WRITE,1,7,NCPW,MESS,30,NCPW)
  GO TO 50
40  CONTINUE
  CALL SIRPAK(MESS,LMS,4HK ,14HWHICH RESULT?d)
50  CONTINUE
  ITEM MENUIN(MLNURL,NITEMS,ITEMS,MESS)
  GO TO (100,200,300,99999),ITEM
C
C SET LOCAL ALL OPTION.
C
100 CONTINUE
  LOCALI=.TRUE.
C
C DETERMINE WHICH OPERATION TO PERFORM ON THE VOLUME VALUE AND
C BRANCH ACCORDINGLY.
C
200 CONTINUE
  GO TO (210,220,230),IOFLAG
C
C READ VOLUME.
C
210 CONTINUE
  LOGVAL=RA1LDR(V,LOCALI,VGOT,
1    MTERSE,IMODE,NCPW,
2    VOLNAM,MX1OG1,CNVFVM,CNVFVA,UNVOL,.FALSE.,
3    .TRUE.,PMES,VMORGN,
4    INKFTL,ANSFRM,
5    NDEFV,DEFALV)
  IF (LOCALI) GO TO 300
  IF (.NOT.AIFLG) GO TO 5
  AIFLG=.FALSE.
  GO TO 99999
C
C EDIT VOLUME.
C
220 CONTINUE
  LOGVAL=RARED1(V,LOCALI,

```

```

VAN01480
VAN01490
VAN01500
VAN01510
VAN01520
VAN01530
VAN01540
VAN01550
VAN01560
VAN01570
VAN01580
VAN01590
VAN01600
VAN01610
VAN01620
VAN01630
VAN01640
VAN01650
VAN01660
VAN01670
VAN01680
VAN01690
VAN01700
VAN01710
VAN01720
VAN01730
VAN01740
VAN01750
VAN01760
VAN01770
VAN01780
VAN01790
VAN01800
VAN01810
VAN01820
VAN01830
VAN01840
VAN01850
VAN01860
VAN01870
VAN01880
VAN01890
VAN01900
VAN01910
VAN01920
VAN01930
VAN01940
VAN01950
VAN01960

```



```

1      MIERSE, NCPW,
2      VOLNAM, NFROM, NTO,
3      CNVFVM, CNVFA, UNVOL,
4      .TRUE., PMES, VMORGN,
5      RNRFIL, ANSFRM,
6      NDEFV, DEFALV)
      IF (LOCAL) GO TO 300
      IF (.NOT. ALLFLG) GO TO 5
      ALLFLG=.FALSE.
      GO TO 99999

C
C WRITE VOLUME.
C
230 CONTINUE
      LOGVAL=RARDMP(LOCAL,
1      MIERSE, OMODI, NCPW,
2      V, VOLNAM, FROM, VGOI,
3      CNVFVM, CNVFA, UNVOL,
4      .TRUE., PMES, VMORGN,
5      RNRFIL, ANSFRM)
      IF (LOCAL) GO TO 300
      IF (.NOT. ALLFLG) GO TO 5
      ALLFLG=.FALSE.
      GO TO 99999

C
C DETERMINE WHICH OPERATION TO PERFORM ON THE WEIGHT VALUE AND
C BRANCH ACCORDINGLY.
C
300 CONTINUE
      GO TO (310, 320, 330), IOFLAG

C READ WEIGHT.
C
310 CONTINUE
      LOGVAL=RAT1DR(WT, LOCAL, WTGOI,
1      MIERSE, IMODE, NCPW,
2      WETNAM, MXTOGI, CNVFVM, CNVFA, NAMF12, .FALSE.,
3      .TRUE., PMES, WTMGRN,
4      RNRFIL, ANSFRM,
5      NDEFWT, DFALWT)
      IF (LOCAL) GO TO 99999
      IF (.NOT. ALLFLG) GO TO 5
      ALLFLG=.FALSE.
      GO TO 99999

C EDIT WEIGHT.
C
320 CONTINUE
      LOGVAL=RAREDIT(WI, LOCAL,

```

```

VAN01970
VAN01980
VAN01990
VAN02000
VAN02010
VAN02020
VAN02030
VAN02040
VAN02050
VAN02060
VAN02070
VAN02080
VAN02090
VAN02100
VAN02110
VAN02120
VAN02130
VAN02140
VAN02150
VAN02160
VAN02170
VAN02180
VAN02190
VAN02200
VAN02210
VAN02220
VAN02230
VAN02240
VAN02250
VAN02260
VAN02270
VAN02280
VAN02290
VAN02300
VAN02310
VAN02320
VAN02330
VAN02340
VAN02350
VAN02360
VAN02370
VAN02380
VAN02390
VAN02400
VAN02410
VAN02420
VAN02430
VAN02440
VAN02450

```



```

1      MIERSE, NCPW,
2      WEINAM, NFROM, N10,
3      CONVFM, CONVFA, NAMF12,
4      .TRUE., PMES, WIMRGN,
5      RNWFL, ANSFRM,
6      NDLEW1, DIALW1)
      IF (LOCAL) GO TO 99999
      IF (.NOT.ALFLG) GO TO 5
      ALLFG=.FALSE.
      GO TO 99999

C
C WRITE WLIGHT.
C
330 CONTINUE
      LOGVAL=RAIRDP(LOCAL,
1      MIERSE, OMODI, NCPW,
2      WI, WEINAM, IFROM, WTGOT,
3      CONVFM, CONVFA, NAMF12,
4      .TRUE., PMES, WIMRGN,
5      RNWFL, ANSFRM)
      IF (LOCAL) GO TO 99999
      IF (.NOT.ALFLG) GO TO 5
      ALLFG=.FALSE.

C
C RETURN TO CALLING PROGRAM.
C
99999 CONTINUE
      RETURN
      END

```

```

VAN02460
VAN02470
VAN02480
VAN02490
VAN02500
VAN02510
VAN02520
VAN02530
VAN02540
VAN02550
VAN02560
VAN02570
VAN02580
VAN02590
VAN02600
VAN02610
VAN02620
VAN02630
VAN02640
VAN02650
VAN02660
VAN02670
VAN02680
VAN02690
VAN02700
VAN02710
VAN02720
VAN02730
VAN02740

```



```

C-----CUBE MODULE SUBPROGRAM-----BLO00010
      BLOCK DATA                                BLO00020
C-----SUBPROGRAM DESCRIPTION-----BLO00030
C THIS SUBPROGRAM INITIALIZES VARIABLES IN THE LABELED COMMON BLOCKS OF
C THIS MODULE.                                BLO00040
C EACH LABELED COMMON AND ALL RELATED STATEMENTS AND DEFINITIONS ARE
C LISTED UNDER THE SUBPROGRAM NAME WHERE THE LABELED COMMON FIRST
C APPEARS.                                BLO00050
C-----PROGRAMMERS AND REPORT-----BLO00060
C PROGRAM VERSION: 1                        BLO00070
C PROGRAM DATE : JUNE 1981                 BLO00080
C PROGRAMMERS : C. CHRYSSOSTOMIDIS AND R. CELOTOIO BLO00090
C REPORT : IDIAS ON HOW TO WRITE DEX MODULES BLO00100
C : VOLUME AND WEIGHT OF A S.W. PARALLELPiped BLO00110
C : CUBE MODULE ROUTINES BLO00120
C REPORT NUMBER : 81-1 BLO00130
C AUTHORS : C. CHRYSSOSTOMIDIS AND R. CELOTOIO BLO00140
C PUBLISHER : MASSACHUSETTS INSTITUTE OF TECHNOLOGY BLO00150
C : DEPARTMENT OF OCEAN ENGINEERING BLO00160
C : DESIGN LABORATORY BLO00170
C : CAMBRIDGE MASS 02139, JUNE 1981 BLO00180
C-----SUBROUTINE MAINPG-----BLO00190
C.....COMMON/RELINOS/RNRFTL,RNWFIL BLO00200
      COMMON/INOUTT/IMODE,OMODE BLO00210
      COMMON/DIATGF/MIERSE BLO00220
      COMMON/MDNCPW/NCPW BLO00230
      INTEGER RNRI TL,RNWFIL BLO00240
      INTEGER IMODE,OMODE BLO00250
      INTEGER NCPW BLO00260
      LOGICAL MIERSE BLO00270
      DATA RNRI TL/3/ BLO00280
      DATA RNWFIL/4/ BLO00290
      DATA IMODE /2/ BLO00300
      DATA OMODE /2/ BLO00310
      DATA MIERSE/.FALSE./ BLO00320
C**** START OF SITE DEPENDENT CODE BLO00330
      DATA NCPW /4/ BLO00340
C**** END OF SITE DEPENDENT CODE BLO00350
C.....SUBROUTINE UNIT. BLO00360
      COMMON /LUNITS/ PSTLUN,UTOLUN BLO00370
      COMMON /LUNIO/ DBLUNN,DBLUNC,LUNFRM,DEFLUN BLO00380
      INTEGER PSTLUN,UTOLUN BLO00390
      INTEGER DBLUNN(2),DBLUNC(16),LUNFRM(2),DEFLUN BLO00400
      DATA PSTLUN/7/ BLO00410
      DATA UTOLUN/7/ BLO00420
      DATA DBLUNN/4HUIOL,4HUN / BLO00430
      DATA DBLUNC/4HLENG,4HIII U,4HNI T ,4HITO B, BLO00440
      DATA DBLUNN(2),DBLUNC(16),LUNFRM(2),DEFLUN BLO00450
      4HE US,4HED D,4HURIN,4HIC IN, BLO00460
      1 BLO00470
      BLO00480
      BLO00490

```



```

2      4HPUT ,4HOUTP,4HUTK,4H /
3      4H ,4H / ,4H /
DATA LUNFRM/4H(110,4H) /
DATA DEFLN/7/
C.....SUBROUTINE FUNIT.....
COMMON /FUNITS/ PSTUN,UIOFUN
COMMON /IUNFO/ DBTUNN,DBTUNC,TUNFRM,DEFTUN
INTEGER PSTUN,UIOFUN
INTEGER DBTUNN(2),DBTUNC(16),FUNFRM(2),DEFTUN
DATA PSTUN/1/
DATA UIOFUN/1/
DATA DBTUNN/4HUIOT,4HUN /
DATA DBTUNC/4HIME,4H U,4HINIT,4HIO B,
1      4HE US,4HED D,4HURIN,4HIG IN,
2      4HPUT ,4HOUTP,4HUTK,4H /
3      4H ,4H / ,4H /
DATA LUNFRM/4H(110,4H) /
DATA DEFTUN/1/
C.....SUBROUTINE FUNIT.....
COMMON /FUNITS/ PSTUN,UIOFUN
COMMON /FUNITO/ DBFUNN,DBFUNC,FUNFRM,DEFFUN
INTEGER PSTUN,UIOFUN
INTEGER DBFUNN(2),DBFUNC(16),FUNFRM(2),DEFFUN
DATA PSTUN/1/
DATA UIOFUN/1/
DATA DBFUNN/4HUIOF,4HUN /
DATA DBFUNC/4HIFORC,4HE U,4HINIT,4HIO B,
1      4HE US,4HED D,4HURIN,4HIG IN,
2      4HPUT ,4HOUTP,4HUTK,4H /
3      4H ,4H / ,4H /
DATA LUNFRM/4H(110,4H) /
DATA DEFFUN/7/
C.....SUBROUTINE AUNIT.....
COMMON /AUNIT/ PSTAUN,UIOAUN
COMMON /AUNFO/ DBAUNN,DBAUNC,AUNFRM,DEFAUN
INTEGER PSTAUN,UIOAUN
INTEGER DBAUNN(2),DBAUNC(16),AUNFRM(2),DEFAUN
DATA PSTAUN/1/
DATA UIOAUN/1/
DATA DBAUNN/4HUIOA,4HUN /
DATA DBAUNC/4HIANGL,4HE U,4HINIT,4HIO B,
1      4HE US,4HED D,4HURIN,4HIG IN,
2      4HPUT ,4HOUTP,4HUTK,4H /
3      4H ,4H / ,4H /
DATA LUNFRM/4H(110,4H) /
DATA DEFAUN/1/
C.....SUBROUTINE IPUNIT.....
COMMON /IPUNTS/ PSTPUN,UIOIPU
COMMON /IPINFO/ DBTPUN,DBTPUC,TPUFRM,DEFTPU

```

```

BL000500
BL000510
BL000520
BL000530
BL000540
BL000550
BL000560
BL000570
BL000580
BL000590
BL000600
BL000610
BL000620
BL000630
BL000640
BL000650
BL000660
BL000670
BL000680
BL000690
BL000700
BL000710
BL000720
BL000730
BL000740
BL000750
BL000760
BL000770
BL000780
BL000790
BL000800
BL000810
BL000820
BL000830
BL000840
BL000850
BL000860
BL000870
BL000880
BL000890
BL000900
BL000910
BL000920
BL000930
BL000940
BL000950
BL000960
BL000970
BL000980

```



```

INTEGER PSTPUN, UIOTPU
INTEGER DBIPUN(2), DBTPUC(16), TPUFRM(2), DEFTPU
DATA PSTPUN/1/
DATA UIOTPU/1/
DATA DBTPUN/HHULOT,4HPU /
DATA DBTPUC/HHTEMP,4HHERAT,4HHURE ,4HHUNIT,
1 4H TO ,4HBE U,4HSED ,4HIDURI,
2 4HNG 1,4HINPUT,4H OUT,4HPUTS /
3 4H ,4H ,4H /
DATA TPUFRM/HH(110,4H) /
DATA DEFTPU/1/
C..... SUBROUTINE DIMENS.....
COMMON /LINFO/ L, LENAM, LMORGN, DEFALL
COMMON /WINFO/ W, WIDNAM, WMORGN, DEFALW
COMMON /HINFO/ H(4), HEINAM, HMORGN, NDEFH, DEFALH(4)
COMMON /DIMFRM/ LWFRM, HFRM
INTEGER LENAM(2), WIDNAM(2), HEINAM(2)
INTEGER LMORGN(16), WMORGN(16), HMORGN(16)
INTEGER LWFRM(2), HFRM(3), NDEFH
REAL L, W, H, DEFALL, DEFALW, DEFALH
DATA L, W, H(1), H(2), H(3), H(4) /1.0, 1.0, 2.0, 3.0, 4.0/
DATA LENAM/HHLENG,4HHTH /
DATA LMORGN/HHLENG,4HHTH O,4HIF CU,4HBE (,4H????,4H????,4H????,
1 4H)K ,4H ,4H ,4H ,4H ,
2 4H ,4H /
DATA WIDNAM/HHWIDT,4HHH /
DATA WMORGN/HHWIDT,4HHH OF,4H CUB,4HIE (? ,4H????,4H????,4H????),
1 4H)K ,4H ,4H ,4H ,4H ,4H ,
2 4H ,4H /
DATA HEINAM/HHHIEIG,4HHHT /
DATA HMORGN/HHHIEIG,4HHHT O,4HIF CU,4HBE (,4H????,4H????,4H????,
1 4H)K ,4H ,4H ,4H ,4H ,4H ,
2 4H ,4H /
DATA NDEFH/4/
DATA DEFALL, DEFALW, DEFALH(1), DEFALH(2), DEFALH(3), DEFALH(4)
1 /1.1, 2.2, 1., 2., 3., 4./
DATA LWFRM/4H(13,4H.6) /
DATA HFRM /4H(110,4H/4E1,4H3.6)/
C..... SUBROUTINE VANDWT.....
COMMON /VINFO/ V(4), VOLNAM, VMORGN, NDEFV, DEFALV(4)
COMMON /WTINFO/ WT(4), WEINAM, WTMORGN, NDEFWT, DFALWT(4)
COMMON /ANSFRM/ ANSFRM
INTEGER VOLNAM(2), WEINAM(2)
INTEGER VMORGN(16), WTMORGN(16)
INTEGER NDEFV, NDEFWT
INTEGER ANSFRM(3)
REAL V, WT, DEFALV, DFALWT
DATA V(1), V(2), V(3), V(4) /1.0, 2.0, 3.0, 4.0/
DATA WT(1), WT(2), WT(3), WT(4) /1023.2, 2046.4, 3069.6, 4092.8/

```



```

DATA VOLNAM/HHVOLU,HHME /
DATA VMORGN/HHVOLU,HHME O,HHF CU,HHBE (,HH???,HH???,HH???,
1 HHX,HH,HH,HH,HH,HH,
2 HH,HH,HH /
DATA WEINAM/HHWEIG,HHHT /
DATA WTMORGN/HHWEIG,HHHT O,HHF CU,HHBE (,HH???,HH???,HH???,
1 HHX,HH,HH,HH,HH,HH,HH,
2 HH,HH,HH /
DATA NDEFV,NDEFWT/HH,HH/
DATA DEFALV(1),DEFALV(2),DEFALV(3),DEFALV(4)
1 /2,HH2,HH.8HH,7.26,9.68/
DATA DFALWT(1),DFALWT(2),DFALWT(3),DFALWT(4)
1 /2HH76.1,HH952.3,7HH28.H,990HH.6/
DATA ANSFRM/HH(110,HH/HH1,HH3.6)/
END

```

```

BL001480
BL001490
BL001500
BL001510
BL001520
BL001530
BL001540
BL001550
BL001560
BL001570
BL001580
BL001590
BL001600
BL001610
BL001620

```


APPENDIX B

UNIT SUBROUTINE ABBREVIATIONS AND CALLING SEQUENCES

Table B-1. Angle Unit Abbreviations

<u>NAMA03</u>	<u>NAMA06</u>	<u>NAMA08</u>	<u>NAMA12</u>
CYC	CYCLE	CYCLE	CYCLE
RAD	RADIAN	RADIAN	RADIAN
DEG	DEGREE	DEG (ANG)	DEGREE (ANG)
MIN	MINUTE	MIN (ANG)	MINUTE (ANG)
SEC	SECOND	SEC (ANG)	SECOND (ANG)

Table B-2. Force Unit Abbreviations

<u>NAMF02</u>	<u>NAMF03</u>	<u>NAMF12</u>
PL	PDL	POUNDAL
LB	LBF	POUND (FORCE)
ST	ST	SHORT TON
LT	LT	LONG TON
DY	DYN	DYNE
N	N	NEWTON
KP	KGF	KILOPOND

Table B-3. Length Unit Abbreviations

<u>NAML02</u>	<u>NAML06</u>	<u>NAML12</u>
IN	INCH	INCH
FT	FOOT	FOOT
SM	STATMI	STATUTE MILE
NM	NAUTMI	NAUT. MILE
MM	MILLIM	MILLIMETER
CM	CENTIM	CENTIMETER
MT	METER	METER
KM	KILOMT	KILOMETER

Table B-4. Temperature Unit Abbreviations

<u>NAMTP1</u>	<u>NAMTP5</u>	<u>NMTP12</u>
C	DEG-C	DEGREES-C
F	DEG-F	DEGREES-F
K	DEG-K	DEGREES-K
R	DEG-R	DEGREES-R

Table B-5. Time Unit Abbreviations

<u>NAMT02</u>	<u>NAMT03</u>	<u>NAMT06</u>	<u>NAMT12</u>
SC	SEC	SECOND	SECOND
MN	MIN	MINUTE	MINUTE
HR	HR	HOUR	HOUR
DY	DAY	DAY	DAY
WK	WK	WEEK	WEEK
MO	MO	MONTH	MONTH
YR	YR	YEAR	YEAR

Table B-6. Calling Sequences of Derived Units

LOGICAL FUNCTION UAACC (UFAACC,UNAACC,ALLFLG,
CONVA,CONVT,NAMA03,NAMT03,NCPW)

LOGICAL FUNCTION UACCEL(UFACC,UNACC,ALLFLG,
CONVL,CONVT,NAML06,NAMT02,NCPW)

LOGICAL FUNCTION UAREA(UFAREA,UNAREA,ALLFLG,
CONVL,NAML06,NCPW)

LOGICAL FUNCTION UFREQ(UFFREQ,UNFREQ,ALLFLG,
CONVA,CONVT,NAMA08,NAMT03,
UIOAUN,UIOTUN,NCPW)

LOGICAL FUNCTION UKVISC(UFKVIS,UNKVIS,ALLFLG,
CONVL,CONVT,NAML02,NAMT03,
UIOLUN,UIOTUN,NCPW)

LOGICAL FUNCTION UMASS(UFMASS,UNMASS,ALLFLG,
CONVF,CONVL,CONVT,NAMF02,NAML02,
NAMT02,UIOFUN,UIOLUN,UIOTUN,NCPW)

LOGICAL FUNCTION UMPOWR(UFPOWE,UNPOWE,ALLFLG,
CONVF,CONVL,CONVT,NAMF02,NAML02,
NAMT02,UIOFUN,UIOLUN,UIOTUN,NCPW)

LOGICAL FUNCTION UPRESS(UFPRES,UNPRES,ALLFLG,
CONVF,CONVL,NAMF03,NAML02,NCPW)

LOGICAL FUNCTION UPSPEC(UFPSPE,UNPSPE,ALLFLG,
CONVL,CONVT,NAML02,NAMT03,NCPW)

LOGICAL FUNCTION URHO(UFRHO,UNRHO,ALLFLG,
CONVF,CONVL,CONVT,NAMF03,NAML02,
NAMT02,UIOFUN,UIOLUN,UIOTUN,NCPW)

LOGICAL FUNCTION USPEED(UFSPEE,UNSPREE,ALLFLG,
CONVL,CONVT,NAML06,NAMT02,
UIOLUN,UIOTUN,NCPW)

LOGICAL FUNCTION UVOL(UFVOL,UNVOL,ALLFLG,
CONVL,NAML06,NCPW)

APPENDIX C

SAMPLE GENERAL DATABASE

Note: This is an edited version of the listing of the database obtained at the terminal. The actual listing of the items is in a random order due to the hashing function employed during the storing of the entries. Group headings also would not appear at the terminal.

\$TITLE: ** GENERAL DATABASE FOR U.S. DDG-2 'CHARLES F. ADAMS' CLASS **

\$ NAME	TYPE	VALUE	COMMENT
HULL CHARACTERISTICS			
LOA	(R)	0.43700E+03	LENGTH OVERALL (FOOT)
LBP	(R)	0.42000E+03	LENGTH BETWEEN PERPENDICULARS (FOOT)
BEAMDWL	(R)	0.46000E+02	MOLDED BEAM AT DESIGN WATERLINE (FOOT)
BEAMMAX	(R)	0.47000E+02	MAXIMUM BEAM (FOOT)
T	(R)	0.15100E+02	MOLDED DRAFT TO KEEL (FOOT)
CP	(R)	**UNDEFINED**	PRISMATIC COEFFICIENT
CX	(R)	**UNDEFINED**	MIDSHIP COEFFICIENT
CB	(R)	0.53100E+00	BLOCK COEFFICIENT
CWP	(R)	0.78200E+00	WATERPLANE COEFFICIENT
LCB	(R)	**UNDEFINED**	LONGITUDINAL CENTER OF BUOYANCY AS FRACTION OF LBP AFT FP
LCF	(R)	**UNDEFINED**	LONGITUDINAL CENTER OF FLOTATION AS FRACTION OF LBP AFT FP
DEP11110	(R)	0.26000E+02	DEPTH AMIDSHIPS AT CENTERLINE (FOOT)
DRAFTSON	(R)	0.24000E+02	DRAFT OF SONAR DOME (FOOT)
DISPLD	(R)	**UNDEFINED**	MOLDED DISPLACEMENT (LONG TON)
DISPLOT	(R)	0.45000E+04	TOTAL DESIGN DISPLACEMENT INCL. SHELL AND APP. (LONG TON)
WETSURF	(R)	**UNDEFINED**	WETTED SURFACE (FOOT **2)
FOCSL	(I)	0	RAISED FORECASTLE (0=NO 1=YES)
PROPULSION AND POWERING			
PPTYP	(I)	2	TYPE OF PROPULSION PLANT (REED)
SHIP	(R)	0.70000E+05	TOTAL INSTALLED SHAFT HORSEPOWER (HP)
NSHAFT	(I)	2	NUMBER OF PROPELLER SHAFTS
NE	(I)	2	NUMBER OF ENGINES
NB	(I)	4	NUMBER OF BOILERS
VSUS	(R)	0.33000E+02	MAXIMUM CONTINUOUS SUSTAINED SPEED (KNOT)
VEND	(R)	0.14000E+02	ENDURANCE SPEED (KNOT)
ENDUR	(R)	0.60000E+04	ENDURANCE RANGE (NAUT. MILE)
PRPTYP	(I)	1	TYPE OF PROPELLER (1=FP 2=GRP)
DPROP	(R)	**UNDEFINED**	PROPELLER DIAMETER (FOOT)
RPM	(R)	**UNDEFINED**	PROPELLER RPM AT FULL POWER (RPM)
SSFTYP	(I)	1	TYPE OF PRIMARY SHIP SERVICE ELECTRICAL PLANT (REED)
NSSG	(I)	4	NUMBER OF PRIMARY SHIP SERVICE GENERATORS
KWSSER	(R)	0.20000E+04	INSTALLED PRIMARY SHIP SERVICE GENERATOR CAPACITY (KW)
EMETYP	(A)	ARRAY(953)	TYPE OF SECONDARY OR EMERGENCY ELECTRICAL PLANT (REED)
NEMG	(A)	ARRAY(934)	NUMBER OF SECONDARY OR EMERGENCY GENERATORS OF EACH TYPE
KWEMER	(R)	0.20000E+03	INSTALLED EMERGENCY OR SECONDARY GENERATOR CAPACITY (KW)
KWPTMG	(A)	ARRAY(915)	CAPACITY PER SECONDARY OR EMERGENCY GENERATORS (KW)

TRANSVERSE AND DIRECTIONAL STABILITY

GM	(R)	0.55000E+01	METACENTRIC HEIGHT UNCORRECTED (FOOT)
ISURFCOR	(R)	**UNDEFINED**	1 REF SURFACE CORRECTION (FOOT)
CI	(R)	**UNDEFINED**	WATERPLANE MOMENT OF INERTIA COEFFICIENT
IINSIBL	(I)	0	1 IN STABILIZERS (0=NO 1=YES)
NRUDDER	(I)	2	2 NUMBER OF RUDDERS

WEAPONS PAYLOAD

TYPGUNS	(A)	ARRAY(879)	TYPE OF GUNS (REED)
NGUNS	(A)	ARRAY(859)	NUMBER OF GUNS OF EACH TYPE
TYPMSL	(I)	131	TYPE OF MISSILE LAUNCHERS (REED)
NMSL	(I)	1	NUMBER OF MISSILE LAUNCHERS
TYPCLWS	(I)	0	TYPE OF CLOSE-IN WEAPON SYSTEM (REED)
NCIWS	(I)	0	NUMBER OF CLOSE-IN WEAPON SYSTEMS
TYPBPDMS	(I)	0	TYPE OF BASIC POINT DEFENSE MISSILE SYSTEM (REED)
NBPDMS	(I)	0	NUMBER OF BASIC POINT DEFENSE MISSILE LAUNCHERS
TYPTORPL	(I)	180	TYPE OF TORPEDO LAUNCHERS (REED)
NTORPL	(I)	2	NUMBER OF TORPEDO LAUNCHERS
TYPASWL	(I)	148	TYPE OF ASW WEAPON LAUNCHERS (REED)
NASWL	(I)	1	NUMBER OF ASW LAUNCHERS

ELECTRONICS, FIRE CONTROL AND SENSORS

TYPSONAR	(A)	ARRAY(628)	TYPE OF SONAR SYSTEMS (REED)
TYPDOME	(I)	62	TYPE OF SONAR DOME (REED)
TYPSURAD	(I)	9	TYPE OF SURFACE SEARCH RADAR (REED)
TYP3DAIR	(I)	15	TYPE OF 3-D AIR SEARCH RADAR (REED)
TYP2DAIR	(I)	14	TYPE OF 2-D AIR SEARCH RADAR (REED)
TYPGRAD	(I)	**UNDEFINED**	TYPE OF GUN FIRE CONTROL RADARS OR DIRECTORS (REED)
NGRAD	(I)	1	NUMBER OF GUN FIRE CONTROL RADARS OR DIRECTORS
TYPMRAD	(I)	300	TYPE OF MISSILE FIRE CONTROL RADARS OR DIRECTORS
NMSIRAD	(I)	2	NUMBER OF MISSILE FIRE CONTROL RADARS OR DIRECTORS (RFED)
TYPICSG	(I)	**UNDEFINED**	TYPE OF GUN FIRE CONTROL SYSTEM (REED)
TYPICSM	(I)	110	TYPE OF MISSILE FIRE CONTROL SYSTEM (REED)
TYPASWIC	(I)	184	TYPE OF ASW FIRE CONTROL SYSTEM (REED)
TYPIDS	(I)	0	TYPE OF TACTICAL DATA SYSTEM (REED)

AVIATION CAPABILITY

TYPIELO	(I)	0	TYPE OF HELICOPTERS CARRIED (REED)
NIELO	(I)	0	NUMBER OF HELICOPTERS CARRIED
IIIFR	(I)	0	HELICOPTER IN-FLIGHT REFUELLING CAPABILITY (1=YES 0=NO)

COMPLEMENT

```

NOFF      (1)      20  NUMBER OF SHIP'S OFFICERS
NCPO      (1)      17  NUMBER OF CHIEF PETTY OFFICERS IN SHIP'S CREW
NENICRPM  (1)      302 NUMBER OF ENLISTED IN SHIP'S CREW
NLAGOIF   (1)      4   NUMBER OF OFFICERS ON FLAG STAFF
NENLSTE   (1)      11  NUMBER OF ENLISTED ON FLAG STAFF
NITROOPS  (1)      0   NUMBER OF TROOPS
  
```

GENERAL MASS PROPERTIES

```

TYPMATL  (A)      ARRAY( 1)  TYPE OF MATERIAL FOR HULL AND SUPERSTRUCTURE RESPECTIVELY (REED)
WEIGHT17 (A)      ARRAY(   )  WEIGHTS OF WEIGHT GROUPS 1-7 RESPECTIVELY(LONG TON )
VCG17    (A)      ARRAY(   )  VERTICAL CENTERS OF GRAVITY OF WT. GROUPS 1-7 RESPECTIVELY (FOOT )
WTLOADS  (R)      0.12521E+04  WEIGHT OF GROUP 8 LOADS (LONG TON )
VCGLOADS (R)      0.99000E+01  VERTICAL CENTER OF GRAVITY OF GROUP 8 LOADS (FOOT )
  
```

THE ARRAYS

```

$NAME: TYPMATL , LENGTH= 2 ( 1)
$ 0.10000E+01 0.20000E+01

$NAME: TYPSONAR, LENGTH= 2 ( 628)
$ 0.19000E+02 0.0

$NAME: NCUNS , LENGTH= 3 ( 859)
$ 0.10000E+01 0.10000E+01 0.0

$NAME: TYPGUNS , LENGTH= 3 ( 879)
$ 0.93000E+02 0.94000E+02 0.0

$NAME: KWPEMG , LENGTH= 2 ( 915)
$ 0.10000E+03 0.0

$NAME: NEMG , LENGTH= 2 ( 934)
$ 0.20000E+01 0.0

$NAME: EMETYP , LENGTH= 2 ( 953)
$ 0.10000E+01 0.0
  
```


\$NAME: WEIGHT17,	LENGTH=	7		
\$	0.12243E+04	0.83120E+03		
\$	0.25400E+03	0.25810E+03	0.12250E+03	0.17590E+03
\$NAME: VCG17	, LENGTH=	7		
\$	0.17600E+02	0.13200E+02		
\$	0.25900E+02	0.34800E+02	0.19000E+02	0.26700E+02
				0.20000E+02

APPENDIX D

GENERAL DATABASE ENTRY CODES

TABLE D1

PAYLOAD SHOPPING LIST

ITEM	DEFINITION	ITEM	DEFINITION
<u>Radio Communications</u>			
1	FF (Non ASW Command and Cont)	26	SPS 53
2	FF (ASW Command and Control)	27	SPS 55 W/IFF
3	DDG	28	SPS 48 E
4	Baseline Strike Cruiser	29	SPS 58
5	CG/CGN	30	SPN 6
6	DD 963	31	SPN 10
		32	SPN 12
		33	SPN 35
		34	SPN 42
<u>Radars</u>			
7	Target Acquisition System Mod 0	35	Beacon Video Processor
8	Target Acquisition System Mod 3	36	SPH 2 (RAVIR Mod 3)
9	SPS 10	37	SPS 58C
10	SPS 10 W/IFF	38	SPS 55
11	SPS 26	39	IFF
12	SPS 29		
13	SPS 30	<u>Sonars</u>	
14	SPS 37	40	SQS 53 (DD 963)
15	SPS 39A	41	SQS 56
16	SPS 40	42	SQS 23 (TRAM)
17	SPS 40 W/IFF	43	SQQ 23
18	SPS 40A W/IFF	44	SQS 26 (CX)
19	SPS 43	45	EDD 610E
20	SPS 43 W/IFF	46	SQA 13 (VDS)
21	SPS 48 (V)	47	SQS 35 (IVDS)
22	SPS 48 C	48	UQN 4
23	SPS 49	49	SQS 23
24	SPS 49 W/IFF	50	TACTLASS
25	SPS 52B	51	Escort Towed Array System (ETAS)
		52	SQS 38 (keel)

SOURCE: Michael Reed, "Ship Synthesis Model for Naval Surface Ships" (O.E. and S.S. Thesis, Massachusetts Institute of Technology, 1975), Table 21.

TABLE D1 (Continued)

ITEM	DEFINITION	ITEM	DEFINITION
53	505 Sonar	<u>Electronic Tactical Data Systems</u>	
54	VDS	73	Conc C and C-FF
55	SQR-19 Towed Array	74	Conv C and C-DD
<u>Sonar Domes</u>		75	ASW C and C-FF-2C, 7D
56	23 Keel Dome (Double, Rubber)	76	NTDS - 4C, 13D - DG/DGN
57	23 Bow Dome	77	NTDS - 2C, 9D - DDG
58	26/53 Bow Dome	78	NTDS - 3C, 15 - 18D
59	610E/505 Bow Dome	79	NTDS - 3C, 15D - CG
60	23 Keel Dome	80	NTDS - 3C, 15D - CGN
61	610E/505 Keel Dome	81	SSCDS - 1C
62	23 Keel Dome (Rubber)	82	Conv C and C - CG, CGN
63	Blank	83	LDG C & C
<u>Sonar Liquid</u>		<u>Guns and Ammo Handling and Stowage</u>	
64	Sonar Liquid (1 ton)	84	3/50 SM, MK-34 w/Shield Mn DK
<u>Electronic Countermeasure (ECM)</u>		85	3/50 SM, MK-34 w/Shield 01 LV
65	FF/FFG Basic	86	3/50 SM, MK-34 w/o Shield 01 LV
66	DD/DDG/CG/CGN Basic (ECM & IR Decoy)	87	3/50 TM, MK-33 w/Shield Mn DK
67	SLQ 32 (V2)	88	3/50 TM, MK-33 w/Shield 01 LV
68	ALR-59	89	3/50 TM, MK-33 w/o Shield Mn DK
69	WLR-8	90	3/50 TM, MK-33 w/o Shield 01 LV
70	SLR-10 or BRD-4, SLR-11	91	5/38 SM, MK-30 Mn DK
71	DTP Suite Three	92	5/38 SM, MK-30 01 LV
72	Nixie	93	5/54 SM, MK-42 Mn DK
		94	5/54 SM, MK-42 01 LV
		95	5/54 SM, LW, MK-45/0 Mn DK
		96	5/54 SM, LW, MK-45/0 01 LV
		97	8/55 SM, MK-71 Mod X Mn DK
		98	8/55 SM, MK-71 Mod X 01 LV

TABLE D1 (Continued)

TABLE D1 (Continued)

ITEM	DEFINITION	ITEM	DEFINITION
146	Tarpon Canister Launcher w/4 missiles)	<u>Missile & Rocket Ammo</u>	
147	Super RBOC (2 launchers)	(n)	
<u>ASW Rocket and Missile Launchers</u>		167	Tomahawk & Cannisters for VLS (n)
148	ASROC-8, MK-16/4 w/8 ASROC, 01 level	168	Tomahawk (non VLS) (n)
149	ASROC-8, MK-16/4 w/8 ASROC, 02 level	169	RIM 67A-Ter 2 DK
150	ASROC Reload, Mag & Handling Gear	170	SM-2, ASROC, Harpoon & Cannister for VLS (n)
151	DASH Launch (Hangar)	171	RIM 7H Sparrow
152	Blank	172	5" Rocket, MK-10
<u>Missile Fire Control System</u>		173	CHAFFROC
153	Near Term Laser System Less Pointer Trainer	174	Harpoon, ASROC, SM-2 (not VLS) (n)
154	Tomahawk WCS MK 2	175	or Tartar (n)
155	Weapons Direction System MK-13		Subroc (2 launcher ready service) (n)
156	MK-74 FCS (1 DIR) (Tartar C)	<u>Torpedo Tubes, Handling & Stowage</u>	
157	MK-74 FCS (2 DIR)	176	MK-25 TT 2 DK
158	MK-115 BPDMS FCS	177	MK-25 TT 01 LV
159	CHAFFROC FCS	178	MK-32 Twin Mn DK
160	MK-99 FCS (2 chan, for SPY-1B)	179	MK-32 Twin 01 LV (r)
161	MK-91 GMFCS (2 chan)	180	MK-32 Triple
162	AEGIS SPY-1	181	Torpedo Countermeasure Launchers
163	AEGIS SPY-1A	182	Hedgehog
164	AEGIS SPY-1B	183	MK-32 Triple, 1 mount (in hull) (n)
165	Harpoon FCS	<u>ASW and Tropedo Fire Control Systems</u>	
166	Tartar D FCS (2 chan)	184	MK-114 UBFCFS
		185	MK-114/12 UBFCFS
		186	MK-116 UBFCFS (r)
		187	Drone Control SRW4C w/URW 15

TABLE D1 (Continued)

<u>ITEM</u>	<u>DEFINITION</u>	<u>ITEM</u>	<u>DEFINITION</u>
188	Drone Control SRW4C	211	LAHV 2
189	TORP FCS (MK-25) (Hull)	212	LAMPS (Platform & Refuel)
190	TORP FCS (MK-25) (Supstr)	213	One LAMPS III Hangar
191	Tarpon FCS Interface Alterations	214	Helo Tie-Down
192	MK-309 UBFCs	215	Aero Stores (1000 ft ³)
		216	Blank
			(n)
<u>ASW Ammo</u>			
193	ASROC Reload (02 Level)		<u>Aviation Liquids</u>
194	ASROC Reload (01 Level)	217	JP 5 (100 gal.)
195	MK-37/0,3 2 DK	218	Av Gas (100 gal.)
196	MK-37/0,3 01 LV	219	Av Lube Oil (100 gal.)
197	MK-37/1,2 2 DK	220	Blank
198	MK-44 Mn DK		<u>Small Arms</u>
199	MK-44 01 LV	221	Misc. - FF/FFG
200	MK-46 Mn DK	222	Misc. - DD/DDG/CG
201	MK-48 Mn DK	223	Misc. - Cruiser
202	MK-48 2 DK	224	Misc. - (Marine Detach.)
203	MK-46 in hull	225	Misc. - Small Ships
<u>Helos</u>			<u>Aviation Ammo</u>
204	Two LAMPS III (main deck)		
205	HASP-T Helo (main deck)	226	MK-46 Torp
206	One SH-3	227	Blank
207	One SH-2D	228	Blank
		229	Blank
<u>Helo Support</u>			
208	LAMPS MK III Package		
209	LAMPS Control		
210	LAHV 1		

TABLE D1 (Continued)

<u>ITEM</u>	<u>DEFINITION</u>	<u>ITEM</u>	<u>DEFINITION</u>
<u>Sewage Treatment Plant</u>		250	50' Utility
230	Sewage Treatment Plant (175 man)	251	Blank
231	Blank	<u>Future Development</u>	
<u>Replenishment at Sea & Cargo Handling</u>		252	Weight Margin (1 ton)
232	FAST System - Dest/Cruiser	<u>Armor</u>	
233	Blank	253	Armor for GMLS
<u>Liquids</u>		254	LVL II Armor for one 64 cell VLS (n)
234	NSFO (100 gal.)	255	LVL II Armor for Torp Mag & EOS (n)
235	Diesel Oil (100 gal.)	256	LVL II CIC Anti-frag armor (n)
236	Potable Water (100 gal.)	257	LVL II Fwd & Aft Aegis (n)
237	Liquid O ₂ (100 gal.)	258	Frag Protection SPS-49 Room & Ill. Base
238	Auto Gas ² (100 gal.)	259	Armor for one 5" or 8" gun (n)
239	Blank	<u>Undefined Items</u>	
<u>Boats</u>		260	Blank
240	14' Punt	thru	Blank
241	26' Motor Whale Boat (Wherry)	300	
242	26' Personnel	300+	See TABLE D-2
243	28' Personnel		
244	33' Utility		
245	33' Personnel		
246	35' Motor Boat		
247	40' Utility		
248	40' Personnel		
249	50' Motor Launch		

n: new item - 6 December 1979

r: updated item - 6 December 1979

TABLE D-2

SUPPLEMENTAL PAYLOAD SHOPPING LIST

<u>ITEM</u>	<u>DEFINITION</u>
<u>Fire Control Radars and Directors</u>	
300	SPG-51C
301	SPG-55
302	SPG-60
303	MK-51 Gun Fire Control Director
304	MK-63 Gun Fire Control Director
<u>Missile Launchers and Fire Control Systems</u>	
305	Tartar MK-11 GMLS
306	Tartar MK-22 GMLS
307	MK-11 MFCS
308	MK-76 MFCS

TABLE D-3

INDEX TO NON-PAYLOAD REED CODE

<u>ITEM</u>	<u>CODE</u>	<u>MEANING</u>
Propulsion Plant Type (PPTYP)	1	600 psi steam
	2	1200 psi steam
	3	1200 psi pressure-fired steam
	4	nuclear
	5	gas turbine first generation
	6	gas turbine second generation
	7	diesel
	8	COGAS
Ship Service Electric Plant Type (SSEPTYP)	1	steam
	2	gas turbine first generation
	3	gas turbine second generation
	4	low speed diesel
	5	medium speed diesel
	6	high speed diesel
Emergency Electric Plant Type (EMETYP)	1	gas turbine first generation
	2	gas turbine second generation
	3	low speed diesel
	4	medium speed diesel
	5	high speed diesel
Type of Material (TYPMATL)	1	steel
	2	aluminum

SOURCE: Michael Reed, "Ship Synthesis Model for Naval Surface Ships" (O.E. and S.M. Thesis, Massachusetts Institute of Technology, 1975).

APPENDIX E

SAMPLE WEIGHT AND VERTICAL CENTER OF GRAVITY DATABASES

Note: These are edited versions of the listings of the databases obtained at the terminal. The actual listings of the items in each database is in a random order due to the hashing function employed during the storing of the entries.

\$ NAME ** WLIGHT DATABASE FOR U. S. DDG-2 'CHARLES F. ADAMS' CLASS **
 VALUE COMMENT

W100	(R)	0.33980E+03	WT. OF SHELL PLATING (LONG TON)
W101	(R)	0.11400E+03	WT. OF LONG. AND TRANS. FRAMING (LONG TON)
W103	(R)	0.89400E+02	WT. OF PLATFORMS AND FLATS (LONG TON)
W107	(R)	0.19260E+03	WT. OF ALL DECKS (LONG TON)
W111	(R)	0.11270E+03	WT. OF SUPERSTRUCTURE (LONG TON)
W112	(R)	0.47600E+02	WT. OF PROPULSION FOUNDATIONS (LONG TON)
W113	(R)	0.71200E+02	WT. OF FDN FOR AUX. AND OTHER EQUIP (LONG TON)
W114	(R)	0.13020E+03	WT. OF STRUCTURAL BIDS (LONG TON)
W115	(R)	0.21200E+02	WT. OF TRUNKS AND ENCLOSURES (LONG TON)
W119	(R)	0.35200E+02	WT. OF CASTINGS AND FORGINGS (LONG TON)
W120	(R)	0.13000E+02	WT. OF SEA CHESTS (LONG TON)
W123	(R)	0.12700E+02	WT. OF DOORS, HATCHES, SCUTILES (LONG TON)
W125	(R)	0.61000E+01	WT. OF MASIS AND KINGPOSTS (LONG TON)
W127	(R)	0.86000E+01	WT. OF SONAR DOMES (LONG TON)
W150	(R)	0.30000E+02	WT. OF WELDING, RIVETING AND FASTENINGS (LONG TON)
W200	(R)	0.22930E+03	WT. OF BOILERS AND ENERGY CONVERTERS (LONG TON)
W201	(R)	0.12820E+03	WT. OF PROPULSION UNITS (LONG TON)
W202	(R)	0.35200E+02	WT. OF MAIN CONDENSERS AND AIR EJECTORS (LONG TON)
W203	(R)	0.12760E+03	WT. OF PROPELLER, SHAFTING AND BEARINGS (LONG TON)
W204	(R)	0.24100E+02	WT. OF COMBUSTION AIR SUPPLY (LONG TON)
W205	(R)	0.16500E+02	WT. OF UPTAKES AND SMOKEPIPES (LONG TON)
W206	(R)	0.86000E+01	WT. OF PROPULSION CONTROL EQUIPMENT (LONG TON)
W207	(R)	0.64100E+02	WT. OF MAIN STEAM SYSTEM (LONG TON)
W208	(R)	0.62800E+02	WT. OF FEEDWATER AND CONDENSATE SYSTEMS (LONG TON)
W209	(R)	0.17100E+02	WT. OF CIRC. AND CLG. WATER SYSTEMS (LONG TON)
W210	(R)	0.15300E+02	WT. OF F. O. SERVICE SYSTEM (LONG TON)
W211	(R)	0.16900E+02	WT. OF LUBE OIL SYSTEM (LONG TON)
W250	(R)	0.13500E+02	WT. OF PROPULSION REPAIR PARTS (LONG TON)
W251	(R)	0.72000E+02	WT. OF PROPULSION OPERATING FLUIDS (LONG TON)
W300	(R)	0.51800E+02	WT. OF ELECTRICAL POWER GENERATION (LONG TON)
W301	(R)	0.10800E+02	WT. OF POWER DISTRIBUTION SWITCHBOARDS (LONG TON)
W302	(R)	0.40600E+02	WT. OF POWER DISTRIBUTION SYSTEM (LONG TON)
W303	(R)	0.14900E+02	WT. OF LIGHTING SYSTEM (LONG TON)
W350	(R)	0.44000E+01	WT. OF ELECTRICAL PLANT REPAIR PARTS (LONG TON)
W400	(R)	0.14000E+01	WT. OF NAVIGATIONAL EQUIPMENT (LONG TON)
W401	(R)	0.29200E+02	WT. OF I. C. SYSTEMS (LONG TON)
W402	(R)	0.57000E+02	WT. OF GFC'S (LONG TON)
W403	(R)	0.13400E+02	WT. OF COUNTERMEASURES (NON-ELECTRONIC) (LONG TON)
W404	(R)	0.67300E+02	WT. OF ELECTRONIC COUNTERMEASURES (LONG TON)
W406	(R)	**UNDEFINED**	WT. OF ASW FC AND TORPEDO FC SYSTEMS (LONG TON)
W450	(R)	0.76000E+01	WT. OF COMM. AND CONT. REPAIR PARTS (LONG TON)

WEIGHT DATABASE FOR DDG-2 CLASS (CONTINUED)

W500	(R)	0.47000E+01	WT. OF HEATING SYSTEMS (LONG TON)
W501	(R)	0.47200E+02	WT. OF VENTILATION SYSTEMS (LONG TON)
W502	(R)	0.21100E+02	WT. OF AIR COND. SYSTEMS (LONG TON)
W503	(R)	0.76000E+01	WT. OF RETR. SPACES, PLANT AND EQUIP. (LONG TON)
W504	(R)	0.20000E+00	WT. OF GAS, HEAF, AV. L. O., SEWAGE SYSTEMS (LONG TON)
W505	(R)	0.13700E+02	WT. OF PLUMBING INST. (LONG TON)
W506	(R)	0.34400E+02	WT. OF FIREMAIN, FLSING, SPRKLR, S.W. SVC SYSTEMS (LONG TON)
W507	(R)	0.96000E+01	WT. OF FIRE EXT. SYSTEMS (LONG TON)
W508	(R)	0.11300E+02	WT. OF DRNG, BAFLAST, STBLZG TK SYSTEMS (LONG TON)
W509	(R)	0.64000E+01	WT. OF FRESH WATER SYSTEMS (LONG TON)
W510	(R)	0.15000E+01	WT. OF SCUPPERS AND DECK DRAINS (LONG TON)
W511	(R)	0.24900E+02	WT. OF F.O. AND D.O. FILL, VENT, STWG AND XFR SYS (LONG TON)
W512	(R)	0.37000E+01	WT. OF TANK HEATING SYSTEMS (LONG TON)
W513	(R)	0.84000E+01	WT. OF COMPRESSED AIR SYSTEMS (LONG TON)
W514	(R)	0.68300E+02	WT. OF AUX. STEAM, EXH. SIFAM AND STEAM DRAINS (LONG TON)
W517	(R)	0.13300E+02	WT. OF DISTILLING PLANT (LONG TON)
W518	(R)	0.11300E+02	WT. OF SIFERING SYSTEMS (LONG TON)
W519	(R)	0.26300E+02	WT. OF RUDDERS (LONG TON)
W520	(R)	0.28900E+02	WT. OF MOORING, TANG, ANCH AND AC HNDLG, DK MCHRY (LONG TON)
W521	(R)	0.92000E+01	WT. OF ELEV., STORES HANDLING SYSTEMS (LONG TON)
W550	(R)	0.36000E+01	WT. OF AUXILIARY SYSTEMS REPAIR PARTS (LONG TON)
W551	(R)	0.19300E+02	WT. OF AUXILIARY SYSTEMS OP. FLUIDS (LONG TON)
W600	(R)	0.14600E+02	WT. OF HULL FITTINGS (LONG TON)
W601	(R)	0.22000E+02	WT. OF BOATS, BOAT SWG AND HNDLG (LONG TON)
W602	(R)	0.13000E+01	WT. OF RIGGING AND CANVAS (LONG TON)
W603	(R)	0.22000E+02	WT. OF LADDERS AND GRATINGS (LONG TON)
W604	(R)	0.22700E+02	WT. OF NON-STRUCTURAL BHDS AND DOORS (LONG TON)
W605	(R)	0.36300E+02	WT. OF PAINTING (LONG TON)
W606	(R)	0.14100E+02	WT. OF DECK COVERING (LONG TON)
W607	(R)	0.26400E+02	WT. OF HULL INSULATION (LONG TON)
W608	(R)	0.24600E+02	WT. OF STOREROOMS, STOWAGES AND LOCKERS (LONG TON)
W609	(R)	0.41000E+01	WT. OF EQUIP FOR UTILITY SPACES (LONG TON)
W610	(R)	0.10000E+02	WT. OF EQUIP FOR WKSHPs, LABS AND TEST AREAS (LONG TON)
W611	(R)	0.11900E+02	WT. OF EQUIP. FOR GALLEY, PTY, SCLRY AND COMSRY (LONG TON)
W612	(R)	0.31700E+02	WT. OF FURNISHINGS FOR LIVING SPACES (LONG TON)
W613	(R)	0.93000E+01	WT. OF FURNISHINGS FOR OFFICES, CONT. CTRS (LONG TON)
W614	(R)	0.13000E+01	WT. OF FURNISHINGS FOR MED. AND DENTAL SP. (LONG TON)
W650	(R)	0.18000E+01	WT. OF OUTFIT AND FRNSHING REPAIR PARTS (LONG TON)
W700	(R)	0.23380E+03	WT. OF GUNS AND GUNMOUNTS (LONG TON)
W701	(R)	0.76000E+01	WT. OF AMMUNITION HANDLING SYSTEMS (LONG TON)
W702	(R)	0.82000E+01	WT. OF AMMUNITION STOWAGE (LONG TON)

WEIGHT DATABASE FOR DDG-2 CLASS (CONTINUED)

W704	(R)	**UNDEFINED**	WT. OF MISSILE LAUNCHERS (SEE W700) (LONG TON)
W705	(R)	**UNDEFINED**	WT. OF ASW ROCKET LAUNCHERS (SEE W700) (LONG TON)
W750	(R)	0.47000E+01	WT. OF ARMAMENT REPAIR PARTS (LONG TON)
W751	(R)	0.39000E+01	WT. OF ARMAMENT OPERATING FLUIDS (LONG TON)
W800	(R)	0.39100E+02	WT. OF SHIPS OFFICERS, CREW AND EFFECTS (LONG TON)
W803	(R)	0.96300E+02	WT. OF SHIPS AMMUNITION (LONG TON)
W806	(R)	0.58900E+02	WT. OF PROVISIONS AND PERSONNEL STORES (LONG TON)
W812	(R)	0.52700E+02	WT. OF POTABLE WATER (LONG TON)
W813	(R)	0.77300E+02	WT. OF RESERVE FEEDWATER (LONG TON)
W814	(R)	0.15800E+02	WT. OF SHIPS LUBE OIL (LONG TON)
W816	(R)	0.87050E+03	WT. OF FUEL OIL (LONG TON)
W817	(R)	0.41500E+02	WT. OF DIESEL OIL (LONG TON)

STITLE: ** VCG DATABASE FOR U.S. DDG-2 'CHARLES F. ADAMS' CLASS **
 \$ NAME TYPE VALUE COMMENT

VCG100	(R)	0.10500E+02	VCG OF SHELL PLATING (FOOT)
VCG101	(R)	0.10700E+02	VCG OF LONG. AND TRANS. FRAMING (FOOT)
VCG103	(R)	0.16200E+02	VCG OF PLATFORMS AND FLATS (FOOT)
VCG107	(R)	0.27200E+02	VCG OF ALL DECKS (FOOT)
VCG111	(R)	0.38100E+02	VCG OF SUPERSTRUCTURE (FOOT)
VCG112	(R)	0.53000E+01	VCG OF PROPULSION FOUNDATIONS (FOOT)
VCG113	(R)	0.18100E+02	VCG OF FDN FOR AUX. AND OTHER EQUIP (FOOT)
VCG114	(R)	0.14500E+02	VCG OF STRUCTURAL BIDS (FOOT)
VCG115	(R)	0.15700E+02	VCG OF TRUNKS AND ENCLOSURES (FOOT)
VCG119	(R)	0.11500E+02	VCG OF CASTINGS AND FORGINGS (FOOT)
VCG120	(R)	0.38000E+01	VCG OF SEA CHESTS (FOOT)
VCG123	(R)	0.25500E+02	VCG OF DOORS, HATCHES, SCUTILES (FOOT)
VCG125	(R)	0.82000E+02	VCG OF MASTS AND KINGPOSTS (FOOT)
VCG127	(R)	-0.40000E+01	VCG OF SONAR DOMES (FOOT)
VCG150	(R)	0.17600E+02	VCG OF WELDING, RIVETING AND FASTENINGS (FOOT)
VCG200	(R)	0.13400E+02	VCG OF BOILERS AND ENERGY CONVERTERS (FOOT)
VCG201	(R)	0.13300E+02	VCG OF PROPULSION UNITS (FOOT)
VCG202	(R)	0.92000E+01	VCG OF MAIN CONDENSERS AND AIR EJECTORS (FOOT)
VCG203	(R)	0.54000E+01	VCG OF PROPELLERS, SHAFTING AND BEARINGS (FOOT)
VCG204	(R)	0.22200E+02	VCG OF COMBUSTION AIR SUPPLY (FOOT)
VCG205	(R)	0.52600E+02	VCG OF UPLAKES AND SMOKEPIPS (FOOT)
VCG206	(R)	0.16000E+02	VCG OF PROPULSION CONTROL EQUIPMENT (FOOT)
VCG207	(R)	0.20400E+02	VCG OF MAIN STEAM SYSTEM (FOOT)
VCG208	(R)	0.14600E+02	VCG OF FEEDWATER AND CONDENSATE SYSTEMS (FOOT)
VCG209	(R)	0.78000E+01	VCG OF CIRC. AND CLG. WATER SYSTEMS (FOOT)
VCG210	(R)	0.11400E+02	VCG OF E. O. SERVICE SYSTEM (FOOT)
VCG211	(R)	0.82000E+01	VCG OF LUBE OIL SYSTEM (FOOT)
VCG250	(R)	0.19500E+02	VCG OF PROPULSION REPAIR PARTS (FOOT)
VCG251	(R)	0.10600E+02	VCG OF PROPULSION OPERATING FLUIDS (FOOT)
VCG300	(R)	0.16900E+02	VCG OF ELECTRICAL POWER GENERATION (FOOT)
VCG301	(R)	0.18900E+02	VCG OF POWER DISTRIBUTION SWITCHBOARDS (FOOT)
VCG302	(R)	0.20000E+02	VCG OF POWER DISTRIBUTION SYSTEM (FOOT)
VCG303	(R)	0.25100E+02	VCG OF LIGHTING SYSTEM (FOOT)
VCG350	(R)	0.14400E+02	VCG OF ELECTRICAL PLANT REPAIR PARTS (FOOT)
VCG400	(R)	0.53600E+02	VCG OF NAVIGATIONAL EQUIPMENT (FOOT)
VCG401	(R)	0.18300E+02	VCG OF I. C. SYSTEMS (FOOT)
VCG402	(R)	0.37100E+02	VCG OF GUN FIRE CONTROL SYSTEMS (FOOT)
VCG403	(R)	0.19700E+02	VCG OF COUNTERMEASURES (NON-ELECTRONIC) (FOOT)
VCG404	(R)	0.23500E+02	VCG OF ELECTRONIC COUNTERMEASURES (FOOT)
VCG406	(R)	**UNDEFINED**	VCG OF ASW FC AND TORPEDO FC SYSTEMS (FOOT)
VCG450	(R)	0.16300E+02	VCG OF COMM. AND CONT. REPAIR PARTS (FOOT)

VCG DATABASE FOR DDG-2 CLASS (CONTINUED)

VCG500	(R)	0.25200E+02	VCG OF HEATING SYSILMS (FOOT)
VCG501	(R)	0.30900E+02	VCG OF VENTILATION SYSTEMS (FOOT)
VCG502	(R)	0.18400E+02	VCG OF AIR COND. SYSTEMS (FOOT)
VCG503	(R)	0.18600E+02	VCG OF GAS, SPACES, PLANT AND EQUIP. (FOOT)
VCG504	(R)	0.12400E+02	VCG OF GAS, HEAF, AV. L.O., SEWAGE SYSTEMS (FOOT)
VCG505	(R)	0.26400E+02	VCG OF PLUMBING INST. (FOOT)
VCG506	(R)	0.16800E+02	VCG OF FIREMAIN, FLSING, SPRKIR, S.W. SVC SYSTEMS (FOOT)
VCG507	(R)	0.22900E+02	VCG OF FIRE EXT. SYSTEMS (FOOT)
VCG508	(R)	0.10600E+02	VCG OF DRNG, BALLAST, STBLZG TK SYSTEMS (FOOT)
VCG509	(R)	0.21900E+02	VCG OF FRSH WATER SYSTEM (FOOT)
VCG510	(R)	0.37400E+02	VCG OF SCUPPERS AND DECK DRAINS (FOOT)
VCG511	(R)	0.11200E+02	VCG OF F.O. AND D.O. FILL, VENT, STWG AND XFR SYS (FOOT)
VCG512	(R)	0.62000E+01	VCG OF TANK HEATING SYSTEMS (FOOT)
VCG513	(R)	0.20300E+02	VCG OF COMPRESSED AIR SYSTEMS (FOOT)
VCG514	(R)	0.18100E+02	VCG OF AUX. SIAM, EXH. SIAM AND STEAM DRAINS (FOOT)
VCG517	(R)	0.17300E+02	VCG OF DISTILLING PLANI (FOOT)
VCG518	(R)	0.18000E+02	VCG OF STEERING SYSTEMS (FOOT)
VCG519	(R)	0.13100E+02	VCG OF RUDDERS (FOOT)
VCG520	(R)	0.24000E+02	VCG OF MOORING, IVNG, ANCH AND AC HNDLG, DK MCHRY (FOOT)
VCG521	(R)	0.38500E+02	VCG OF ELEV., STORES HANDLING SYSTEMS (FOOT)
VCG550	(R)	0.19800E+02	VCG OF AUXILIARY SYSTEMS REPAIR PARTS (FOOT)
VCG551	(R)	0.16900E+02	VCG OF AUXILIARY SYSTEMS OP. FLUIDS (FOOT)
VCG600	(R)	0.32500E+02	VCG OF HULL FITTINGS (FOOT)
VCG601	(R)	0.40200E+02	VCG OF BOATS, BOAT STWG AND HNDLG (FOOT)
VCG602	(R)	0.37600E+02	VCG OF RIGGING AND CANVAS (FOOT)
VCG603	(R)	0.13200E+02	VCG OF LADDERS AND GRATINGS (FOOT)
VCG604	(R)	0.28800E+02	VCG OF NON-STRUCTURAL BIIDS AND DOORS (FOOT)
VCG605	(R)	0.20000E+02	VCG OF PAINTING (FOOT)
VCG606	(R)	0.26700E+02	VCG OF DECK COVERING (FOOT)
VCG607	(R)	0.28400E+02	VCG OF HULL INSULATION (FOOT)
VCG608	(R)	0.20700E+02	VCG OF STOREROOMS, STOWAGES AND LOCKERS (FOOT)
VCG609	(R)	0.21400E+02	VCG OF EQUIP FOR UTILITY SPACES (FOOT)
VCG610	(R)	0.23800E+02	VCG OF EQUIP FOR WKSHPs, LABS AND TEST AREAS (FOOT)
VCG611	(R)	0.29700E+02	VCG OF EQUIP. FOR GALLEY, PTY, SCLRY AND COMSRY (FOOT)
VCG612	(R)	0.26700E+02	VCG OF FURNISHINGS FOR LIVING SPACES (FOOT)
VCG613	(R)	0.30600E+02	VCG OF FRNSINGS FOR OFFICES, CONT. CIRS. (FOOT)
VCG614	(R)	0.29700E+02	VCG OF FURNISHINGS FOR MED. AND DENTAL SP. (FOOT)
VCG650	(R)	0.15000E+02	VCG OF OUTFIT AND FURNISHING REPAIR PARTS (FOOT)
VCG700	(R)	0.35800E+02	VCG OF GUNS AND GUNMOUNTS (FOOT)
VCG701	(R)	0.33900E+02	VCG OF AMMUNITION HANDLING SYSILMS (FOOT)
VCG702	(R)	0.19600E+02	VCG OF AMMUNITION STORAGE (FOOT)

VCG DATABASE FOR DDG-2 CLASS (CONTINUED)

VCG704	(R)	**UNDEFINED**	VCG OF MISSILE LAUNCHERS (SEE VCG700) (FOOT)
VCG705	(R)	**UNDEFINED**	VCG OF ASW ROCKET LAUNCHERS (SEE VCG700) (FOOT)
VCG750	(R)	0.13500E+02	VCG OF ARMAMENT REPAIR PARTS (FOOT)
VCG751	(R)	0.33700E+02	VCG OF ARMAMENT OPERATING FLUIDS (FOOT)
VCG800	(R)	0.23500E+02	VCG OF SHIPS OFFICERS, CREW AND EFFECTS (FOOT)
VCG803	(R)	0.17200E+02	VCG OF SHIPS AMMUNITION (FOOT)
VCG806	(R)	0.16800E+02	VCG OF PROVISIONS AND PERSONNEL STORES (FOOT)
VCG812	(R)	0.44000E+01	VCG OF POTABLE WATER (FOOT)
VCG813	(R)	0.54000E+01	VCG OF RESERVE FEEDWATER (FOOT)
VCG814	(R)	0.17400E+02	VCG OF SHIPS LUBE OIL (FOOT)
VCG816	(R)	0.87000E+01	VCG OF FUEL OIL (FOOT)
VCG817	(R)	0.93000E+01	VCG OF DIESEL OIL (FOOT)

APPENDIX F

MACHWT MODULE LISTING

Note: Subroutines MAINPG and MODIO are included in the Cube Module listing in Appendix A and do not appear here.


```

C-----MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM-----MW100010
C      SUBROUTINE INPUT(CALALL,IOFLAG)MW100020
C-----SUBPROGRAM DESCRIPTION-----MW100030
C      SUBROUTINE INPUT PROVIDES THE USER WITH A MENU FROM WHICH TO
C      CHOOSE WHICH MODULE SEGMENT IT IS DESIRED TO OPERATE NEXT. THE
C      CHOICES ARE:MW100040
C      ALL MODULE INPUT VARIABLES MW100050
C      THE MODULE UNITS TO BE USED DURING INPUT AND OUTPUT MW100060
C      THE MACHINERY WEIGHT ITEM TO BE ESTIMATED. MW100070
C      THE DATA FROM EXISTING SHIPS TO BE USED FOR CURVE FITTING MW100080
C      FOR W(200) AND W(201) MW100090
C      THE CHARACTERISTICS OF THE NEW SHIP DESIGN NEEDED AS INPUT MW100100
C      TO ALLOW THE CALCULATIONS TO BE DONE MW100110
C      THE UNITS MODULE ALLOWS THE USER TO SPECIFY THE LENGTH, FORCE MW100120
C      AND TIME UNITS TO BE USED DURING INPUT AND OUTPUT. LENGTH UNITS ARE MW100130
C      NEEDED FOR LBP, PROPELLER DIAMETER, DRAIT AND POSSIBLY SPEED. THESE MW100140
C      ARE ALL USED IN CALCULATING W(203). TIME UNITS MAY BE NEEDED FOR MW100150
C      SPEED. FORCE UNITS ARE NEEDED FOR WEIGHTS. MW100160
C      THE MACHINERY WEIGHT ITEM NAME SHOULD BE READ BEFORE OTHER INPUT MW100170
C      IS ENTERED. MW100180
C      IN READING DATA FROM EXISTING DESIGNS FOR CURVE FITTING, THE USER MW100190
C      IS ASKED TO SPECIFY WHICH SEQUENTIAL NUMBER OF DATA POINTS THIS PAIR MW100200
C      REPRESENTS. MW100210
C-----SUBPROGRAM ASSUMPTIONS-----MW100220
C      NONF YFI MW100230
C-----INPUT VARIABLES-----MW100240
C      CALALL: .TRUE. IF THE INPUT VALUE OF CALALL WAS .TRUE. AND NO ERROR MW100250
C      OCCURRED WHEN READING OR EDITING AN ESSENTIAL VARI MW100260
C      ABLE MW100270
C      .FALSE. IF THE INPUT VALUE OF CALALL WAS .FALSE. OR AN ERROR MW100280
C      OCCURRED WHEN READING OR EDITING AN ESSENTIAL VARI MW100290
C      ABLE MW100300
C-----INPUT VARIABLES-----MW100310
C      CALALL: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE MW100320
C      .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE MW100330
C      IOFLAG: DENOTES THE OPERATION TO BE PERFORMED MW100340
C      =1 IF THE USER WISHES TO READ THE VARIABLES MW100350
C      2 EDIT MW100360
C      3 WRITE MW100370
C-----LABELED COMMON VARIABLES-----MW100380
C      LABELED COMMON DIALGF AND MDNCPW HAVE BEEN DEFINED IN SUBROUTINE MW100390
C      MAINPG. MW100400
C      .....WFLAG INITIALIZED IN BLOCK DATA MW100410
C      WFLAG: DENOTES WHICH WEIGHT ITEM IS TO BE ESTIMATED MW100420
C      =1 FOR W(200) MW100430
C      =2 W(201) MW100440
C      =3 W(203) MW100450
C      .....CRVPTS INITIALIZED IN BLOCK DATA MW100460
C      CRVPTS: THE NUMBER OF PAIRS OF POINTS TO BE FITTED MW100470
C      NPTS: THE NUMBER OF PAIRS OF POINTS TO BE FITTED MW100480
C      MW100490

```


C IND	: AN ARRAY WHICH STORES THE VALUES OF THE INDEPENDENT VARIABLE	MM100500
C	FOR CURVE FITTING	MM100510
C	DEP	MM100520
C	: AN ARRAY WHICH STORES THE VALUES OF THE DEPENDENT VARIABLE	MM100530
C	FOR CURVE FITTING	MM100540
C	-----SUBPROGRAMS AND FUNCTIONS CALLED-----	MM100550
C	DEX	MM100560
C	INIT	MM100570
C	IMOVIC	MM100580
C	MINUTIN	MM100590
C	MISOUT	MM100600
C	SIRPAK	MM100610
C	STIRPAK	MM100620
C	DEX LIBRARY	MM100630
C	NONE	MM100640
C	MODULI	MM100650
C	MVCHRT	MM100660
C	MVUNIT	MM100670
C	MVLIST	MM100680
C	LABELED COMMONS	MM100690
C	COMMON /DIALGF/ MTERSE	MM100700
C	COMMON /MDNCPW/ NCPW	MM100710
C	COMMON /WFLAG/ WFLAG	MM100720
C	COMMON /CRVPTS/ NPTS, IND(10), DEP(10)	MM100730
C	VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS	MM100740
C	INTEGER IOTLAC, NCPW	MM100750
C	INTEGER MENUM(2), NITEMS, ITEMS(12), ITEM	MM100760
C	INTEGER MESS(16), LMS	MM100770
C	INTEGER NPTS, NUPT, WFLAG	MM100780
C	INTEGER READ(2), EDIT(2), WRITE(2)	MM100790
C	LOGICAL CALALL, LOCAL	MM100800
C	LOGICAL MTERSE, LOGVAL, LMOVEC, INTIN	MM100810
C	REAL IND, DEP	MM100820
C	VARIABLE DATA DEFINITIONS	MM100830
C	DATA LMS/16/	MM100840
C	DATA MENUM/4HINPU,4HT /	MM100850
C	DATA NITEMS/6/	MM100860
C	DATA ITEMS/4HALL,4H ,	MM100870
C	1 4HUNIT,4HIS ,	MM100880
C	2 4HWT,1,4HITEM ,	MM100890
C	3 4HICURV,4HFEPTS ,	MM100900
C	4 4HNEWS,4HHIP ,	MM100910
C	5 4HIDONE,4H /	MM100920
C	DATA READ /4HREAD,4HLS /	MM100930
C	DATA EDIT /4HEDIT,4HLS /	MM100940
C		MM100950
C		MM100960
C		MM100970
C		MM100980


```

DATA WRITE/HHWRIT,4HE,4/
C
C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF THE LOCAL ALL FLAG IS ACTIVATED HERE MENU 'INPUT' WILL
C NOT BE DEFINED BY THIS INVOCATION OF SUBROUTINE MWINPT.
C
LOCAL=CALALL
IF (LOCAL) GO TO 200
C
C PREPARE A PROMPTING MESSAGE FOR MENU 'INPUT' AND THEN PROVIDE THE
C MENU TO THE USER.
C
5 CONTINUE
IF (MTERSE) GO TO 40
CALL SIRPAK(MESS,LMS,4HK ,40HSELECT WHICH INPUT VARIABLE SEGMENT
1 104)
GO TO (10,20,30),IOFLAG
10 LOGVAL=LMOVEC(READ,1,6,NCPW,MESS,40,NCPW)
GO TO 50
20 LOGVAL=LMOVEC(EDIT,1,6,NCPW,MESS,40,NCPW)
GO TO 50
30 LOGVAL=LMOVEC(WRITE,1,7,NCPW,MESS,40,NCPW)
GO TO 50
40 CALL SIRPAK(MESS,LMS,4HK ,21HWHICH INPUT SEGMENT?)
50 CONTINUE
ITEM MENUIN(MENUIN,NITEMS,11FMS,MESS)
GO TO (100,200,300,400,500,600),ITEM
C
C SET THE INPUT ALL OPTION.
C
100 CONTINUE
LOCAL=.TRUE.
C
C READ, EDIT OR WRITE THE INPUT/INPUT MODULE UNITS.
C
200 CONTINUE
CALL MWUNIT(LOCAL,IOFLAG)
IF (LOCAL) GO TO 300
IF (.NOT.CALALL) GO TO 5
CALALL=.FALSE.
GO TO 600
C
C READ THE WEIGHT ITEM TO BE ESTIMATED.
C
300 CONTINUE
CALL MWLIST
IF (LOCAL) GO TO 400
IF (.NOT.CALALL) GO TO 5
CALALL=.FALSE.

```

MW100990
MW101000
MW101010
MW101020
MW101030
MW101040
MW101050
MW101060
MW101070
MW101080
MW101090
MW101100
MW101110
MW101120
MW101130
MW101140
MW101150
MW101160
MW101170
MW101180
MW101190
MW101200
MW101210
MW101220
MW101230
MW101240
MW101250
MW101260
MW101270
MW101280
MW101290
MW101300
MW101310
MW101320
MW101330
MW101340
MW101350
MW101360
MW101370
MW101380
MW101390
MW101400
MW101410
MW101420
MW101430
MW101440
MW101450
MW101460
MW101470


```

C
C READ, EDIT OR WRITE THE DATA FROM EXISTING SHIPS TO BE USED FOR CURVE
C FITTING. IF THE USER IS INPUTTING DATA, FIRST QUERY THE USER TO
C SPECIFY WHICH SEQUENTIAL PAIR OF DATA POINTS THIS REPRESENTS.
C
400 CONTINUE
  11 (IOFLAG.NE.1) GO TO 450
410 CONTINUE
  CALL SIRPAK(MESS,LMS,4HK ,59HISPECIFY THE SEQUENTIAL NUMBER OF THIS
  11S PAIR OF DATA POINTS.4)
  LOGVAL=LN(IN(1,NEFD,NUPT,MESS))
  11 (LOGVAL) GO TO 440
  CALL SIRPAK(MESS,LMS,4HK ,63HINTIN FAILED WHEN READING IN SEQUEN
  11TIAL NUMBER OF DATA POINTS.4)
  CALL MESOUT(MESS)
  CALL SIRPAK(MESS,LMS,4HK ,49HHHIS INFORMATION IS ESSENTIAL. PLEA
  1S1 TRY AGAIN.4)
  CALL MESOUT(MESS)
  GO TO 50
440 CONTINUE
  11 (NUPT.GT.NPTS) NPTS=NUPT
450 CONTINUE
  CALL MWCHRT(LOCAL,IOFLAG,NUPT)
  11 (LOCAL) GO TO 500
  11 (.NOT.CALALL) GO TO 5
  CALALL=.FALSE.
  GO TO 600
C
C READ, EDIT OR WRITE THE NEW SHIP CHARACTERISTICS. IF INPUTTING DATA,
C PROMPT THE USER AS TO WHICH SUPPLEMENTAL INFORMATION IS NEEDED FOR
C CALCULATING THE PARTICULAR WEIGHT ITEM.
C
500 CONTINUE
  NUPT=0
  11 (IOFLAG.EQ.3) GO TO 550
  11 (WFLAG.NE.3) GO TO 530
  CALL SIRPAK(MESS,LMS,4HK ,58HITO ESTIMATE W(203) THE FOLLOWING IN
  1FORMATION IS REQUIRED:4)
  CALL MESOUT(MESS)
  CALL SIRPAK(MESS,LMS,4HK ,56HILBP PPTYPE SHIP NSHAFT PRPTYP
  1SUS DPROP(OPTIONAL)4)
  CALL MESOUT(MESS)
  GO TO 550
530 CONTINUE
  CALL SIRPAK(MESS,LMS,4HK ,49HITO ESTIMATE W(200) OR W(201) INPUT
  1NLW SHIP SHIP.4)
  CALL MESOUT(MESS)
550 CONTINUE

```

```

MW101480
MW101490
MW101500
MW101510
MW101520
MW101530
MW101540
MW101550
MW101560
MW101570
MW101580
MW101590
MW101600
MW101610
MW101620
MW101630
MW101640
MW101650
MW101660
MW101670
MW101680
MW101690
MW101700
MW101710
MW101720
MW101730
MW101740
MW101750
MW101760
MW101770
MW101780
MW101790
MW101800
MW101810
MW101820
MW101830
MW101840
MW101850
MW101860
MW101870
MW101880
MW101890
MW101900
MW101910
MW101920
MW101930
MW101940
MW101950
MW101960

```



```

CALL MWCHRT(LOCAL,IOFLAG,NUPT)
IF (LOCAL) GO TO 600
IF (.NOT.CALAL) GO TO 5
CALAL=.FALSE.
C RETURN CONTROL TO CALLING PROGRAM.
C 600 CONTINUE
RETURN
END

```

```

MW101970
MW101980
MW101990
MW102000
MW102010
MW102020
MW102030
MW102040
MW102050
MW102060

```



```

C-----MACHINERY WEIGHT MODULE SUBPROGRAM-----MWU00010
C      SUBROUTINE MWUNIT(CALALL,IOFLAG)MWU00020
C-----SUBPROGRAM DESCRIPTION-----MWU00030
C SUBROUTINE MWUNIT ALLOWS ITS USERS TO SELECT WHICH MODULE UNIT THEY
C WISH TO READ, EDIT OR WRITE. THE CHOICES ARE:
C      ALL MODULE UNITS
C      THE LENGTH UNIT
C      THE TIME UNIT
C      THE FORCE UNIT
C-----SUBPROGRAM ASSUMPTIONS-----MWU00090
C IF THE CALLING PROGRAM ALL OPTION IS ACTIVE, THE LOCAL ALL OPTION IS
C SET TO .TRUE. UPON INVOKING THIS SUBROUTINE. IN THIS CASE THE MENU IS
C NOT DEFINED.
C-----OUTPUT VARIABLES-----MWU00130
C CALALL: .TRUE. IF THE INPUT VALUE OF CALALL WAS .TRUE. AND NO ERROR
C OCCURRED IN READING OR EDITING A MODULE UNIT
C      .FALSE. IF THE INPUT VALUE OF CALALL WAS .FALSE. OR AN ERROR
C OCCURRED IN READING OR EDITING A MODULE UNIT
C-----INPUT VARIABLES-----MWU00190
C CALALL: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C      .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE
C IOFLAG: DENOTES THE OPERATION TO BE PERFORMED
C      1 IF THE USER WISHES TO READ MODULE UNITS
C      2      EDIT
C      3      WRITE
C-----LABELLED COMMON VARIABLES-----MWU00260
C LABELLED COMMON DIALGF, INOUTF, MDNCPW AND REFNOS HAVE BEEN DEFINED IN
C SUBROUTINE MAINPG.
C LABELLED COMMONS UNITS, TUNITS, FUNITS, IUNIO, IUNITO AND FUNIO
C TUNIO AND IUNIO HAVE BEEN INITIALIZED IN SUBPROGRAM BLOCK DATA.
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----MWU00310
C DEX
C      STRPAK
C      LMOVIC
C      LNUIN
C DEX LIBRARY
C      LUNIT
C      TUNIT
C      FUNIT
C MODULE
C      NONE
C-----MWU00420
C LABELLED COMMONS
C      COMMON /DIALGF/ MTERSE
C      COMMON /INOUTF/ IMODE,OMODE
C      COMMON /MDNCPW/ NCPW
C      COMMON /REFNOS/ RNRFIL,RNWFIL

```



```

RNFILL=RNRFIL
GO TO 15
10 CONTINUE
LOMODE=OMODE
RNFILL=RNRFIL

C
C ACTIVATE THE LOCAL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF THE LOCAL ALL OPTION IS SET IN THIS MANNER, MENU 'UNIT'
C IS NOT DEFINED BY THE INVOCATION OF THIS SUBROUTINE.
C
15 CONTINUE
LOCAL=CALALL
11 (LOCAL) GO TO 200

C
C PREPARE PROMPTING MESSAGE FOR MENU 'UNIT'.
C
20 11 (MTERSE) GO TO 40
CALL STRPAK(MESS,LMS,40K ,22)SELECT WHICH UNIT TO<
GO TO (25,30,35),10FLAG
25 CONTINUE
LOGVAL=LMOVEC(READ,1,6,NCPW,MESS,22,NCPW)
GO TO 50
30 CONTINUE
LOGVAL=LMOVEC(EDIT,1,6,NCPW,MESS,22,NCPW)
GO TO 50
35 CONTINUE
LOGVAL=LMOVEC(WRITE,1,7,NCPW,MESS,22,NCPW)
GO TO 50
40 CONTINUE
CALL STRPAK(MESS,LMS,40K ,12)WHICH UNIT?4

C
C SELECT AN ITEM FROM MENU 'UNIT' AND BRANCH ACCORDINGLY.
C
50 CONTINUE
ITEM=MENUIN(MENUNM,NITEMS,11LMS,MESS)
GO TO (100,200,300,400,500),ITEM

C
C ACTIVATE THE SUBPROGRAM'S ALL OPTION.
C
100 CONTINUE
LOCAL=.TRUE.

C
C READ, EDIT OR WRITE THE LENGTH UNIT.
C
200 CONTINUE
CALL LUNIT(UTOLUN,LOCAL,
1 TOTAC,LOMODE,MTERSE,NCPW,
2 DBLUNN,DBLUNC,
3 PMPREP,PMES,

```

```

MW000990
MW001000
MW001010
MW001020
MW001030
MW001040
MW001050
MW001060
MW001070
MW001080
MW001090
MW001100
MW001110
MW001120
MW001130
MW001140
MW001150
MW001160
MW001170
MW001180
MW001190
MW001200
MW001210
MW001220
MW001230
MW001240
MW001250
MW001260
MW001270
MW001280
MW001290
MW001300
MW001310
MW001320
MW001330
MW001340
MW001350
MW001360
MW001370
MW001380
MW001390
MW001400
MW001410
MW001420
MW001430
MW001440
MW001450
MW001460
MW001470

```



```

4      RNFILE,LUNFRM,
5      DEFLUN)
      IF (LOCAL) GO TO 300
      IF (.NOT.CALALL) GO TO 20
      CALALL=.FALSE.
      GO TO 500

C READ, EDIT OR WRITE THE TIME UNIT.
C
300 CONTINUE
      CALL TUNIT(UOTUN,LOCAL,
1      IOFLAG,IOMODE,MIERSE,NCPW,
2      DBFUNN,DBFUNC,
3      PMPRIP,PMIES,
4      RNFILE,LUNFRM,
5      DEFTUN)
      IF (LOCAL) GO TO 400
      IF (.NOT.CALALL) GO TO 20
      CALALL=.FALSE.
      GO TO 500

C READ, EDIT OR WRITE THE FORCE UNIT.
C
400 CONTINUE
      CALL TUNIT(UIOFUN,LOCAL,
1      IOFLAG,IOMODE,MIERSE,NCPW,
2      DBFUNN,DBFUNC,
3      PMPRIP,PMIES,
4      RNFILE,LUNFRM,
5      DIFFUN)
      IF (LOCAL) GO TO 500
      IF (.NOT.CALALL) GO TO 20
      CALALL=.FALSE.

C RETURN CONTROL TO THE CALLING PROGRAM.
C
500 CONTINUE
      RETURN
      END

```

```

MWU01480
MWU01490
MWU01500
MWU01510
MWU01520
MWU01530
MWU01540
MWU01550
MWU01560
MWU01570
MWU01580
MWU01590
MWU01600
MWU01610
MWU01620
MWU01630
MWU01640
MWU01650
MWU01660
MWU01670
MWU01680
MWU01690
MWU01700
MWU01710
MWU01720
MWU01730
MWU01740
MWU01750
MWU01760
MWU01770
MWU01780
MWU01790
MWU01800
MWU01810
MWU01820
MWU01830
MWU01840
MWU01850
MWU01860

```



```

C-----MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM-----MWL00010
C      SUBROUTINE MWLIST-----MWL00020
C-----SUBPROGRAM DESCRIPTION-----MWL00030
C      SUBROUTINE MWLIST SETS THE VALUE OF WFLAG WHEN THE USER SPECIFIES
C      ONE OF THE FOLLOWING MACHINERY WEIGHT ITEMS TO BE ESTIMATED.
C      W(200) BOILERS AND ENERGY CONVERTERS
C      W(201) PROPULSION UNITS
C      W(203) PROPELLER, SHAFTING AND BEARINGS
C-----SUBPROGRAM ASSUMPTIONS-----MWL00070
C      NONE YET
C-----Labeled COMMON VARIABLES-----MWL00080
C      LABELLED COMMON DIALCF HAS BEEN DEFINED IN SUBROUTINE MAINPG.
C      LABELLED COMMON WIFLAG HAS BEEN DEFINED IN SUBROUTINE MWINPT.
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----MWL00090
C      DEX
C      SIRPAK
C      MINUTIN
C      DEX LIBRARY
C      NONE
C      MODUFT
C      NONE
C-----MWL00100
C      COMMON /DIALCF/ MIERSE
C      COMMON /WIFLAG/ WFLAG
C-----MWL00110
C      VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C      INTEGER MENUM(2), NITEMS, ITEMS(6), ITEM
C      INTEGER MESS(12), LMS, WFLAG
C      LOGICAL MIERSE
C-----MWL00120
C      VARIABLE DATA DEFINITIONS
C      DATA LMS/12/
C      DATA MENUM/4H1,1,4H1LMS/
C      DATA NITEMS/3/
C      DATA ITEMS/4H200,4H
C      1 4H201,4H
C      2 4H203,4H /
C-----MWL00130
C      PREPARE A PROMPTING MESSAGE FOR MENU 'WT.ITEMS' AND THEN PROVIDE THE
C      MENU.
C      IF (MIERSE) GO TO 10
C      CALL SIRPAK(MESS,LMS,4H ,4HSELECT THE DESIRED WEIGHT ITEM TO BE
C      1E ESTIMATED.4)
C-----MWL00140
C-----MWL00150
C-----MWL00160
C-----MWL00170
C-----MWL00180
C-----MWL00190
C-----MWL00200
C-----MWL00210
C-----MWL00220
C-----MWL00230
C-----MWL00240
C-----MWL00250
C-----MWL00260
C-----MWL00270
C-----MWL00280
C-----MWL00290
C-----MWL00300
C-----MWL00310
C-----MWL00320
C-----MWL00330
C-----MWL00340
C-----MWL00350
C-----MWL00360
C-----MWL00370
C-----MWL00380
C-----MWL00390
C-----MWL00400
C-----MWL00410
C-----MWL00420
C-----MWL00430
C-----MWL00440
C-----MWL00450
C-----MWL00460
C-----MWL00470
C-----MWL00480
C-----MWL00490

```



```

GO TO 50
10 CONTINUE
CALL SIRPAK(MESS,LMS,4HK ,19HWLCH WEIGHT ITEM?4
50 CONTINUE
ITEM MENUIN(MENUNM,NITEMS,ITEMS,MESS)
GO TO (100,200,300),ITEM

C USER WISHES TO ESTIMATE W(200).
100 CONTINUE
WLAG=1
GO TO 99999

C USER WISHES TO ESTIMATE W(200).
C
200 CONTINUE
WLAG=2
GO TO 99999

C USER WISHES TO ESTIMATE W(203).
C
300 CONTINUE
WLAG=3

C RETURN TO CALLING PROGRAM
C
99999 CONTINUE
RETURN
END

MWL00500
MWL00510
MWL00520
MWL00530
MWL00540
MWL00550
MWL00560
MWL00570
MWL00580
MWL00590
MWL00600
MWL00610
MWL00620
MWL00630
MWL00640
MWL00650
MWL00660
MWL00670
MWL00680
MWL00690
MWL00700
MWL00710
MWL00720
MWL00730
MWL00740
MWL00750
MWL00760
MWL00770

```



```

C-----MACHINERY WEIGHT MODULE SUBPROGRAM-----MWC00010
SUBROUTINE MWCHRT(CALALL,IOFLAG,NUPT)MWC00020
C-----SUBPROGRAM DESCRIPTION-----MWC00030
C SUBROUTINE MWCHRT ALLOWS THE USER TO READ, EDIT OR WRITE THE
C FOLLOWING CHARACTERISTICS OF A SHIP DESIGN:MWC00040
C LENGTH BETWEEN PERPENDICULARSMWC00050
C DRAFT (KEEL)MWC00060
C TYPE OF PROPULSION PLANTMWC00070
C INSTALLED SHAFT HORSEPOWERMWC00080
C NUMBER OF PROPELLER SHAFTSMWC00090
C MAXIMUM SUSTAINED SPEEDMWC00100
C TYPE OF PROPELLER (FP OR CRP)MWC00110
C PROPELLER DIAMETERMWC00120
C WEIGHT OF BOILERSMWC00130
C WEIGHT OF PROPULSION UNITSMWC00140
C WEIGHT OF PROPELLER, SHAFTING AND BEARINGS OF THE LATTER 3MWC00150
C FOR A NEW SHIP DESIGN. WHEN OBTAINING DATA FOR CALCULATING PARAMETRICMWC00160
C EQUATIONS FOR ESTIMATING THE WEIGHTS, THE INDEPENDENT VARIABLE MUSTMWC00170
C BE READ FIRST AND THE DEPENDENT VARIABLE SECOND.MWC00180
C-----SUBPROGRAM ASSUMPTIONS-----MWC00190
C THIS SUBROUTINE IS ACCESSED THROUGH EITHER ITEM 'CURVE PT'
C OR ITEM 'NEW SHIP' OF SUBROUTINE MWENP. WHEN READING DATA POINTS
C THAT ARE TO BE USED FOR DETERMINING PARAMETRIC EQUATIONS, CHRCIR
C SHOULD BE ACCESSED VIA 'CURVE PT'. WHEN READING IN SUPPLEMENTAL
C INFORMATION ABOUT THE NEW SHIP DESIGN NEEDED FOR OBTAINING THE PARTI-
C CULAR WEIGHT ESTIMATE, IT SHOULD BE ACCESSED THROUGH 'NEW SHIP'.
C BOTH VALUES OF A PAIR OF DATA POINTS MUST BE OBTAINED ONCE
C CHRCIR HAS BEEN ACCESSED BEFORE RETURNING TO THE CALLING PROGRAM.
C-----OUTPUT VARIABLES-----MWC00200
C CALALL: .TRUE. IF THE INPUT VALUE OF CALALL WAS .TRUE. AND NO ERROR
C OCCURRED WHILE READING OR EDITING A VITAL VARIABLEMWC00210
C .FALSE. IF THE INPUT VALUE OF CALALL WAS .FALSE. OR AN ERROR
C OCCURRED WHILE READING OR EDITING A VITAL VARIABLEMWC00220
C-----INPUT VARIABLES-----MWC00230
C CALALL: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVEMWC00240
C IOFLAG: DENOTES THE OPERATION BEING PERFORMEDMWC00250
C =1 IF THE USER WISHES TO READ THE VARIABLES
C 2 EDITMWC00260
C 3 WRIITEMWC00270
C NUPT : A NUMBER WHICH INDICATES EITHER THE SEQUENTIAL NUMBER OF THE
C DATA POINTS TO BE READ OR THAT NEW SHIP INFORMATION IS
C SOUGHTMWC00280
C-----LABELLED COMMON VARIABLES-----MWC00290
C LABELLED COMMON DIALGF, INOUTF, MDNCPW AND REINOS HAVE BEEN DEFINED IN
C SUBROUTINE MAINPG.MWC00300
C LABELLED COMMONS FUNITS,LUNITS AND TUNITS HAVE BEEN DEFINED IN
C SUBROUTINE MWUNIT.MWC00310

```


C.....VARIRM INITIALIZED IN BLOCK DATA
C INTRM: FORMAT TO BE USED FOR READING INTEGERS FROM OR WRITING INTE-
C GERS TO A SEQUENTIAL FILE
C RSCRM: FORMAT TO BE USED WHEN READING REAL SCALARS FROM OR WRITING
C REAL SCALARS TO A SEQUENTIAL FILE
C.....CRVPTS INITIALIZED IN BLOCK DATA
C NPTS : THE SEQUENTIAL NUMBER OF THE DATA PAIR BEING READ. WHEN THE
C LAST PAIR HAS BEEN ENTERED, NPTS WILL EQUAL THE TOTAL NUMBER
C OF DATA POINTS.
C IND : AN ARRAY CONTAINING THE VALUES OF THE INDEPENDENT VARIABLE TO
C BE USED FOR CURVE FITTING
C DEP : AN ARRAY CONTAINING THE VALUES OF THE DEPENDENT VARIABLE TO BE
C USED FOR CURVE FITTING
C.....INFO2 INITIALIZED IN BLOCK DATA
C LBPNAME: THE DATABASE NAME OF THE LENGTH BETWEEN PERPENDICULARS
C CMNT2 : THE DATABASE COMMENT WHICH DESCRIBES LBPNAME
C DEF2 : THE DEFAULT VALUE FOR LBP
C.....INFO5 INITIALIZED IN BLOCK DATA
C HNAME : THE DATABASE NAME OF THE DRAFT
C CMNT5 : THE DATABASE COMMENT WHICH DESCRIBES HNAME
C DEF5 : THE DEFAULT VALUE FOR DRAFT
C.....INFO19 INITIALIZED IN BLOCK DATA
C PPNAME: THE DATABASE NAME OF THE TYPE OF PROPULSION PLANT
C CMNT19: THE DATABASE COMMENT WHICH DESCRIBES PPNAME
C DEF19 : THE DEFAULT VALUE FOR THE TYPE OF PROPULSION PLANT
C.....INFO20 INITIALIZED IN BLOCK DATA
C SHIPNAME: THE DATABASE NAME OF THE INSTALLED SHAFT HORSEPOWER
C CMNT20: THE DATABASE COMMENT WHICH DESCRIBES SHIPNAME
C DEF20 : THE DEFAULT VALUE FOR INSTALLED SHAFT HORSEPOWER
C.....INFO21 INITIALIZED IN BLOCK DATA
C NSHPNAME: THE DATABASE NAME FOR THE NUMBER OF PROPELLER SHAFTS
C CMNT21: THE DATABASE COMMENT WHICH DESCRIBES NSHPNAME
C DEF21 : THE DEFAULT VALUE FOR THE NUMBER OF PROPELLER SHAFTS
C.....INFO24 INITIALIZED IN BLOCK DATA
C VSUSNAME: THE DATABASE NAME FOR THE MAXIMUM SUSTAINED SPEED
C CMNT24: THE DATABASE COMMENT WHICH DESCRIBES VSUSNAME
C DEF24 : THE DEFAULT VALUE FOR THE MAXIMUM SUSTAINED SPEED
C.....INFO27 INITIALIZED IN BLOCK DATA
C PRPNAME: THE DATABASE NAME FOR THE TYPE OF PROPELLER
C CMNT27: THE DATABASE COMMENT WHICH DESCRIBES PRPNAME
C DEF27 : THE DEFAULT VALUE FOR THE TYPE OF PROPELLER
C.....INFO28 INITIALIZED IN BLOCK DATA
C DPRPNAME: THE DATABASE NAME FOR THE PROPELLER DIAMETER
C CMNT28: THE DATABASE COMMENT WHICH DESCRIBES DPRPNAME
C DEF28 : THE DEFAULT VALUE FOR THE PROPELLER DIAMETER
C.....INFO200 INITIALIZED IN BLOCK DATA
C W200NAME: THE DATABASE NAME FOR THE WEIGHT OF BOILERS AND ENERGY
C CONVERTERS
C W200C : THE DATABASE COMMENT WHICH DESCRIBES W200NAME

MWC00500
MWC00510
MWC00520
MWC00530
MWC00540
MWC00550
MWC00560
MWC00570
MWC00580
MWC00590
MWC00600
MWC00610
MWC00620
MWC00630
MWC00640
MWC00650
MWC00660
MWC00670
MWC00680
MWC00690
MWC00700
MWC00710
MWC00720
MWC00730
MWC00740
MWC00750
MWC00760
MWC00770
MWC00780
MWC00790
MWC00800
MWC00810
MWC00820
MWC00830
MWC00840
MWC00850
MWC00860
MWC00870
MWC00880
MWC00890
MWC00900
MWC00910
MWC00920
MWC00930
MWC00940
MWC00950
MWC00960
MWC00970
MWC00980

C	DEF200:	THE DEFAULT VALUE FOR THE WEIGHT OF BOILERS AND ENERGY	MWC00990
C	CONVERTERS		MWC01000
CINF201	INITIALIZED IN BLOCK DATA	MWC01010
C	W201NH:	THE DATABASE NAME FOR THE WEIGHT OF PROPULSION UNITS	MWC01020
C	W201C :	THE DATABASE COMMENT WHICH DESCRIBES W200NM	MWC01030
C	DEF201:	THE DEFAULT VALUE FOR THE WEIGHT OF PROPULSION UNITS	MWC01040
CINF203	INITIALIZED IN BLOCK DATA	MWC01050
C	W203NH:	THE DATABASE NAME FOR THE WEIGHT OF PROPELLER, SHAFTING	MWC01060
C	AND BEARINGS		MWC01070
C	W203C :	THE DATABASE COMMENT WHICH DESCRIBES W203NM	MWC01080
C	DEF203:	THE DEFAULT VALUE FOR THE WEIGHT OF PROPELLER, SHAFTING	MWC01090
C	AND BEARINGS		MWC01100
CNEWINF	INITIALIZED IN BLOCK DATA	MWC01110
C	LBPNEW:	THE VALUE OF THE LENGTH BETWEEN PERPENDICULARS OF THE NEW SHIP	MWC01120
C	INLEW :	THE VALUE OF THE KEEL DRAFT OF THE NEW SHIP	MWC01130
C	PPTNEW:	THE VALUE OF THE TYPE OF PROPULSION PLANT OF THE NEW SHIP	MWC01140
C	SHIPNEW:	THE VALUE OF THE INSTALLED SHAFT HORSEPOWER OF THE NEW SHIP	MWC01150
C	NSHINEV:	THE VALUE OF THE NUMBER OF PROPELLER SHAFTS OF THE NEW SHIP	MWC01160
C	VSUSNU:	THE VALUE OF THE MAXIMUM SUSTAINED SPEED OF THE NEW SHIP	MWC01170
C	PRINew:	THE VALUE OF THE TYPE OF PROPELLER OF THE NEW SHIP	MWC01180
C	DPRNEW:	THE VALUE OF THE PROPELLER DIAMETER OF THE NEW SHIP	MWC01190
C	W200NU:	THE VALUE OF THE WEIGHT OF THE BOILERS AND ENERGY CONVERTERS	MWC01200
C	OF THE NEW SHIP		MWC01210
C	W201NU:	THE VALUE OF THE WEIGHT OF PROPULSION UNITS OF THE NEW SHIP	MWC01220
C	W203NU:	THE VALUE OF THE WEIGHT OF PROPELLER, SHAFTING AND BEARINGS	MWC01230
C	OF THE NEW SHIP		MWC01240
C	-----SUBPROGRAMS AND FUNCTIONS CALLED-----		MWC01250
C	DEX		MWC01260
C	SIRPAK		MWC01270
C	LHOVIC		MWC01280
C	MLNUIN		MWC01290
C	MF SOUT		MWC01300
C	DEX LIBRARY		MWC01310
C	ISCLDR		MWC01320
C	ISCLDT		MWC01330
C	ISDRIP		MWC01340
C	RSCLDR		MWC01350
C	RSCEDT		MWC01360
C	RSCOMP		MWC01370
C	UNITLF		MWC01380
C	UNITTF		MWC01390
C	MODULI		MWC01400
C	NONE		MWC01410
C			MWC01420
C			MWC01430
C			MWC01440
C			MWC01450
C			MWC01460
C			MWC01470

C	LABELFD	COMMONS
C		COMMON /DIALGT/ MTERSE
C		COMMON /MNCPPW/ NCPW

COMMON	/INOUT11/	IMODE, OMODE	MWC01480
COMMON	/IRFLNOS/	RNRFIL, RNWFL	MWC01490
COMMON	/VARIRM/	INTIRM, RSCFRM	MWC01500
COMMON	/IUNITS/	PSIFUN, UIOFUN	MWC01510
COMMON	/LUNITS/	PSILUN, UIOLUN	MWC01520
COMMON	/IUNITS/	PSITUN, UIOTUN	MWC01530
COMMON	/CRVPS/	NPIS, INDI(10), DEP(10)	MWC01540
COMMON	/INI02/	LBPNAM, CMNT2, DEF2	MWC01550
COMMON	/INI05/	IINAME, CMNT5, DEF5	MWC01560
COMMON	/INI019/	PPINAM, CMNT19, DEF19	MWC01570
COMMON	/INI020/	SHIPNAM, CMNT20, DEF20	MWC01580
COMMON	/INI021/	NSHIFNM, CMNT21, DEF21	MWC01590
COMMON	/INI024/	VSUSNM, CMNT24, DEF24	MWC01600
COMMON	/INI027/	PRPNAM, CMNT27, DEF27	MWC01610
COMMON	/INI028/	DPRPNM, CMNT28, DEF28	MWC01620
COMMON	/INI200/	W200NM, W200C, DEF200	MWC01630
COMMON	/INI201/	W201NM, W201C, DEF201	MWC01640
COMMON	/INI203/	W203NM, W203C, DEF203	MWC01650
COMMON	/NEWINI/	LBPNEW, IINNEW, PPINNEW, SHIPNEW, NSHIFNEW, VSUSNU, PRINNEW,	MWC01660
		DPRNEW, W200NU, W201NU, W203NU	MWC01670
			MWC01680
			MWC01690
			MWC01700
			MWC01710
			MWC01720
			MWC01730
			MWC01740
			MWC01750
			MWC01760
			MWC01770
			MWC01780
			MWC01790
			MWC01800
			MWC01810
			MWC01820
			MWC01830
			MWC01840
			MWC01850
			MWC01860
			MWC01870
			MWC01880
			MWC01890
			MWC01900
			MWC01910
			MWC01920
			MWC01930
			MWC01940
			MWC01950
			MWC01960

C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS

C

INTEGER IOFLAG, IMODE, OMODE, NCPW

INTEGER RNRFIL, RNWFL, INTIRM(2), RSCFRM(2)

INTEGER PSIFUN, UIOFUN, PSILUN, UIOLUN, PSITUN, UIOTUN

INTEGER NUPT

INTEGER IVAR, IDEI, DBNAME(2), PMES(16), PMORGN(16)

INTEGER LBPNAM(2), IINAME(2), PPINAM(2), SHIPNAM(2), NSHIFNM(2),

VSUSNM(2), PRPNAM(2), DPRPNM(2), W200NM(2), W201NM(2),

W203NM(2)

1

2

INTEGER CMNT2(16), CMNT5(16), CMNT19(8), CMNT20(12), CMNT21(7),

CMNT24(16), CMNT27(7), CMNT28(9), W200C(13), W201C(10),

W203C(14)

2

INTEGER DEF19, DEF21, DEF27

INTEGER PPTNEW, NSHIFNEW, PRINNEW

INTEGER MLSS(16), LMS

INTEGER MENUUM(2), NITEMS, ITEMS(24), ITEM

INTEGER MINUYS(2), NITEMY, ITEM(4), YITEM

INTEGER READ(2), EDIT(2), WRITE(2)

INTEGER UNITNM(3), NAMLO2(1), NAMLO6(2), NAME12(3), NAMEF02(1),

NAMF03(1), NAMEF12(3), NAMEI02(1), NAMEI03(1), NAMEI06(2),

NAMT12(3), UNSPEE(3), KNOT(3), UNSHIP(3), UNKW(3)

1

2

LOGICAL CALALL, MTERSE, LOGVAL, LMOVEC

LOGICAL MENUFL, PMPREP, VITAL

LOGICAL ISCLDR, ISCEDT, ISCDMP, RSCLDR, RSCEDT, RSCDMP

LOGICAL UNITTF, UNITLF, UNITTF, USPEED

REAL DEF2, DEF5, DEF20, DEF24, DEF28, DEF200, DEF201, DEF203

REAL LBPNEW, IINNEW, SHIPNEW, VSUSNU, DPRNEW, W200NU, W201NU, W203NU


```

1      PSTUN,U10FUN,NCPW)
C      IF (.NOT.LOGVAL) GO TO 99999
C
C DETERMINE THE NAMES OF THE FORCE UNITS FOR THE WEIGHTS
C OF THE MACHINERY ITEMS AND THE MULTIPLICATIVE CONVERSION FACTOR.
C
      LOGVAL=UNIT11(CONVIM,NAMF02,NAMF03,NAMF12,CALALL,
1      PSTUN,U10FUN,NCPW)
      IF (.NOT.LOGVAL) GO TO 99999
C
C PREPARE A PROMPTING MESSAGE FOR MENU 'CHARACT.' AND PROVIDE IT
C TO USFR.
C
      N=0
5      CONTINUE
      IF (MIFRSE) GO TO 40
      CALL STRPAK(MESS,IMS,4HK ,32)SELECT WHICH CHARACTERISTIC 10<
      GO TO (10,20,30),10FLAG
10     CONTINUE
      LOGVAL=LMOVEC(READ,1,6,NCPW,MESS,32,NCPW)
      GO TO 50
20     CONTINUE
      LOGVAL=LMOVEC(LDIT,1,6,NCPW,MESS,32,NCPW)
      GO TO 50
30     CONTINUE
      LOGVAL=LMOVEC(WRITE,1,7,NCPW,MESS,32,NCPW)
      GO TO 50
40     CONTINUE
      CALL STRPAK(MESS,IMS,4HK ,22)WHICH CHARACTERISTIC?4
50     CONTINUE
      ITEM MENUIN(MENUNM,NITLMS,ITEMS,MESS)
C
C PROVIDE INDICATOR TO TELL IF VARIABLE TO BE READ IS AN INDEPENDENT
C VARIABLE, A DEPENDENT VARIABLE OR A NEW SHIP CHARACTERISTIC.
C
      N=N+1
      IF (NUPF.EQ.0) N=0
      IF (N.EQ.3.AND.ITEM.NE.12) GO TO 8000
      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,99999),ITEM
C
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE LBP.
C
100    CONTINUE
      IF (10FLAG.EQ.3) RVAR=LBPNEW
      DNAME(1)=LBPNAM(1)
      DNAME(2)=LBPNAM(2)
      UNIT11M=CONVLM
      UNIT11A=CONVLA

```

MWC02460
MWC02470
MWC02480
MWC02490
MWC02500
MWC02510
MWC02520
MWC02530
MWC02540
MWC02550
MWC02560
MWC02570
MWC02580
MWC02590
MWC02600
MWC02610
MWC02620
MWC02630
MWC02640
MWC02650
MWC02660
MWC02670
MWC02680
MWC02690
MWC02700
MWC02710
MWC02720
MWC02730
MWC02740
MWC02750
MWC02760
MWC02770
MWC02780
MWC02790
MWC02800
MWC02810
MWC02820
MWC02830
MWC02840
MWC02850
MWC02860
MWC02870
MWC02880
MWC02890
MWC02900
MWC02910
MWC02920
MWC02930
MWC02940


```

UNITNM(1)=NAML12(1)
UNITNM(2)=NAML12(2)
UNITNM(3)=NAML12(3)
DO 110 I=1,16
  PHORGN(1)=CMNT2(1)
110 CONTINUE
  RDEF=DEF2
  GO TO (2100,2200,2300),IOFLAG

C
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE DRAFT.
C
200 CONTINUE
  IF (IOFLAG.EQ.3) RVAR=HNEW
  DBNAME(1)=HNAME(1)
  DBNAME(2)=HNAME(2)
  UNITNM=CONVIM
  UNIT1A=CONVIA
  UNITNM(1)=NAML12(1)
  UNITNM(2)=NAML12(2)
  UNITNM(3)=NAML12(3)
  DO 210 I=1,16
    PHORGN(1)=CMNT5(1)
210 CONTINUE
  RDEF=DEF5
  GO TO (2100,2200,2300),IOFLAG

C
C SUBSTITUTE VARIABLES AND BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE TYPE OF PROPULSION PLANI.
C
300 CONTINUE
  IF (IOFLAG.EQ.3) IVAR=PPTNEW
  DBNAME(1)=PPTNAM(1)
  DBNAME(2)=PPTNAM(2)
  DO 310 I=1,8
    PHORGN(1)=CMNT19(1)
310 CONTINUE
  IDEF=DEF19
  GO TO (3100,3200,3300),IOFLAG

C
C SUBSTITUTE VARIABLES AND BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE INSTALLED SHIP.
C
400 CONTINUE
  IF (IOFLAG.EQ.3) IVAR=SHIPNEW
  DBNAME(1)=SHIPNAM(1)
  DBNAME(2)=SHIPNAM(2)

C
C PREPARE MESSAGE FOR READING, EDITING OR WRITING SHAFT HORSPOWER.

```

MWC02950
MWC02960
MWC02970
MWC02980
MWC02990
MWC03000
MWC03010
MWC03020
MWC03030
MWC03040
MWC03050
MWC03060
MWC03070
MWC03080
MWC03090
MWC03100
MWC03110
MWC03120
MWC03130
MWC03140
MWC03150
MWC03160
MWC03170
MWC03180
MWC03190
MWC03200
MWC03210
MWC03220
MWC03230
MWC03240
MWC03250
MWC03260
MWC03270
MWC03280
MWC03290
MWC03300
MWC03310
MWC03320
MWC03330
MWC03340
MWC03350
MWC03360
MWC03370
MWC03380
MWC03390
MWC03400
MWC03410
MWC03420
MWC03430


```

C          CALL STIRPAK(MESS,LMS,4HK ,54HSHAFTHORSEPOWER MUST BE IN UNITS OMWCO3440
1F 111HR HP OR KW.4 MWCO3450
CALL MESOUT(MESS) MWCO3460
CALL STIRPAK(MESS,LMS,4HK ,38HARE THE UNITS OF SHAFTHORSEPOWER HMWCO3480
1P?4 MWCO3490
YTEM=MENUTN(MENUYS,NITEMY,IITEMY,MESS) MWCO3500
GO TO (410,420),YTEM MWCO3510
C THE 1/0 UNITS OF SHIP ARE HP. MWCO3520
C 410 CONTINUE MWCO3530
UNITIM=1.0 MWCO3540
UNITNM(1)=UNSHIP(1) MWCO3550
UNITNM(2)=UNSHIP(2) MWCO3560
UNITNM(3)=UNSHIP(3) MWCO3570
GO TO 440 MWCO3580
C THE 1/0 UNITS OF SHIP ARE KW. MWCO3590
C 420 CONTINUE MWCO3600
UNITIM=0.0 MWCO3610
DO 450 I=1,12 MWCO3620
PHORGN(I)=CMNT20(I) MWCO3630
RDET=DEF20 MWCO3640
GO TO (2100,2200,2300),IOFLAG MWCO3650
C 430 CONTINUE MWCO3660
UNITNM(1)=UNKW(1) MWCO3670
UNITNM(2)=UNKW(2) MWCO3680
UNITNM(3)=UNKW(3) MWCO3690
440 CONTINUE MWCO3700
UNITIM=0.0 MWCO3710
DO 450 I=1,12 MWCO3720
PHORGN(I)=CMNT20(I) MWCO3730
RDET=DEF20 MWCO3740
GO TO (2100,2200,2300),IOFLAG MWCO3750
C 450 CONTINUE MWCO3760
UNITNM(1)=UNKW(1) MWCO3770
UNITNM(2)=UNKW(2) MWCO3780
UNITNM(3)=UNKW(3) MWCO3790
GO TO 500 MWCO3800
C 500 CONTINUE MWCO3810
CALL STIRPAK(MESS,LMS,4HK ,33HDOES VSUS HAVE UNITS OF 'KNOTS'?4 MWCO3820
YTEM=MENUTN(MENUYS,NITEMY,IITEMY,MESS) MWCO3830
GO TO (510,520),YTEM MWCO3840
C 510 CONTINUE MWCO3850
CALL STIRPAK(MESS,LMS,4HK ,24HTHANK YOU. PROCEEDING.4) MWCO3860
CALL MESOUT(MESS) MWCO3870
UNITNM(1)=KNOT(1) MWCO3880
UNITNM(2)=KNOT(2) MWCO3890
UNITNM(3)=KNOT(3) MWCO3900
UNITIM=1.0 MWCO3910
GO TO 530 MWCO3920

```



```

520 CONTINUE
CALL STIRPAK(MESS,LMS,4HK, .62H THE CONVERSION FACTORS FOR CHANGING
1 THE INPUT/OUTPUT UNITS 104
CALL MESOUT(MESS)
CALL STIRPAK(MESS,LMS,4HK, .64H THE PROGRAM STANDARD UNITS OF KNOTS
1 WILL BE BASED ON THE LENGTH)
CALL MESOUT(MESS)
CALL STIRPAK(MESS,LMS,4HK, .35H AND TIME UNITS YOU HAVE SPECIFIED.
1)
CALL MESOUT(MESS)
LOGVAL=USPLED(USPREE,UNSPREE,CALALL,
1 CONVM,CONVM,NAM106,NAM102,
2 UTOFUN,UTOFUN,NCPW)
IF (.NOT.LOGVAL) GO TO 99999
UNITIM=USPLE/1.688
UNITIM(1)=UNSPREE(1)
UNITIM(2)=UNSPREE(2)
UNITIM(3)=UNSPREE(3)
530 CONTINUE
IF (IOFLAG.EQ.3) RVAR=VSUSNU
DRNAME(1)=VSUSNM(1)
DRNAME(2)=VSUSNM(2)
UNIT1A=0.0
DO 550 I=1,16
PHORGN(1)=CHN124(I)
550 CONTINUE
RDL DEF24
GO TO (2100,2200,2300),IOFLAG
C
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE NUMBER OF PROPELLER SHAFTS.
C
600 CONTINUE
IF (IOFLAG.EQ.3) IVAR=NSHNEW
DRNAME(1)=NSHNM(1)
DRNAME(2)=NSHNM(2)
DO 610 I=1,7
PHORGN(1)=CMNT21(I)
610 CONTINUE
IDF=DEF21
GO TO (3100,3200,3300),IOFLAG
C
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE TYPE OF PROPELLER.
C
700 CONTINUE
IF (IOFLAG.EQ.3) IVAR=PRNEW
DRNAME(1)=PRPNAM(1)
DRNAME(2)=PRPNAM(2)

```



```

DO 710 I=1,7
  PHORGN(1)=CMN127(1)
710 CONTINUE
  IDEF=DEF27
  GO TO (3100,3200,3300),IOFLAG
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE THE PROPELLER DIAMETER.
C
800 CONTINUE
  IF (IOFLAG.EQ.3) RVAR=DPRNEW
  DBNAME(1)=DPRNM(1)
  DBNAME(2)=DPRNM(2)
  UNITIM=CONVIM
  UNITIA=CONVIA
  UNITIM(1)=NAML12(1)
  UNITIM(2)=NAML12(2)
  UNITIM(3)=NAML12(3)
  DO 810 I=1,9
    PHORGN(1)=CMN128(1)
810 CONTINUE
    RDEF=DEF28
    GO TO (2100,2200,2300),IOFLAG
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE W(200).
C
900 CONTINUE
  IF (IOFLAG.EQ.3) RVAR=W200NU
  DBNAME(1)=W200NM(1)
  DBNAME(2)=W200NM(2)
  UNITIM=CONVIM
  UNITIA=CONVIA
  UNITIM(1)=NAMF12(1)
  UNITIM(2)=NAMF12(2)
  UNITIM(3)=NAMF12(3)
  DO 910 I=1,13
    PHORGN(1)=W200C(1)
910 CONTINUE
    RDEF=DEF200
    GO TO (2100,2200,2300),IOFLAG
C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE W(201).
C
1000 CONTINUE
  IF (IOFLAG.EQ.3) RVAR=W201NU
  DBNAME(1)=W201NM(1)
  DBNAME(2)=W201NM(2)

```

MWC04420
MWC04430
MWC04440
MWC04450
MWC04460
MWC04470
MWC04480
MWC04490
MWC04500
MWC04510
MWC04520
MWC04530
MWC04540
MWC04550
MWC04560
MWC04570
MWC04580
MWC04590
MWC04600
MWC04610
MWC04620
MWC04630
MWC04640
MWC04650
MWC04660
MWC04670
MWC04680
MWC04690
MWC04700
MWC04710
MWC04720
MWC04730
MWC04740
MWC04750
MWC04760
MWC04770
MWC04780
MWC04790
MWC04800
MWC04810
MWC04820
MWC04830
MWC04840
MWC04850
MWC04860
MWC04870
MWC04880
MWC04890
MWC04900


```

UNITFM=CONVFM
UNITFA=CONVFA
UNITNM(1)=NAMF 12(1)
UNITNM(2)=NAMF 12(2)
UNITNM(3)=NAMF 12(3)
DO 1010 I=1,10
  PHORGN(I)=W201C(I)
1010 CONTINUE
RDEF=DEF201
GO TO (2100,2200,2300),IOFLAG

C SUBSTITUTE VARIABLES AND THEN BRANCH ACCORDING TO WHETHER TO READ,
C EDIT OR WRITE W(203).
C
1100 CONTINUE
IF (IOFLAG.EQ.3) RVAR=W203NU
DBNAME(1)=W203NM(1)
DBNAME(2)=W203NM(2)
UNITFM=CONVFM
UNITFA=CONVFA
UNITNM(1)=NAMF 12(1)
UNITNM(2)=NAMF 12(2)
UNITNM(3)=NAMF 12(3)
DO 1110 I=1,14
  PHORGN(I)=W203C(I)
1110 CONTINUE
RDEF=DEF203
GO TO (2100,2200,2300),IOFLAG

C READ A REAL VARIABLE.
C
2100 CONTINUE
LOGVAL=RSCIDR(RVAR,CALALL,
1 M1ERSE,IMODE,NCPL,
2 DBNAME,UNITFM,UNITFA,UNITNM,.FALSE.,
3 PMPREP,PMES,PMORGN,
4 RNRFIL,RSCFRM,RDEF)
IF (.NOT.LOGVAL) GO TO 5
IF (NUPT.EQ.0) GO TO 2150
IF (N.EQ.2) GO TO 2125
IND(NUPT)=RVAR
GO TO 5
2125 CONTINUE
DEF(NUPT)=RVAR
GO TO 5
2150 CONTINUE
GO TO (5010,5020,5,5040,5050,5,5,5080,5090,5100,5110,99999),ITEM
C
C EDIT THE REAL VARIABLE.

```

```

MWC04910
MWC04920
MWC04930
MWC04940
MWC04950
MWC04960
MWC04970
MWC04980
MWC04990
MWC05000
MWC05010
MWC05020
MWC05030
MWC05040
MWC05050
MWC05060
MWC05070
MWC05080
MWC05090
MWC05100
MWC05110
MWC05120
MWC05130
MWC05140
MWC05150
MWC05160
MWC05170
MWC05180
MWC05190
MWC05200
MWC05210
MWC05220
MWC05230
MWC05240
MWC05250
MWC05260
MWC05270
MWC05280
MWC05290
MWC05300
MWC05310
MWC05320
MWC05330
MWC05340
MWC05350
MWC05360
MWC05370
MWC05380
MWC05390

```



```

C
2200 CONTINUE
LOGVAL=RSCDIT(IVAR,CALALL,
1 MIERSE,NCPW,
2 DBNAME,UNITFM,UNITFA,UNITNM,
3 PMPREP,PMES,PMORCN)
IF (.NOT.LOGVAL) GO TO 5
IF (NUPT.EQ.0) GO TO 2250
IF (N.EQ.2) GO TO 2225
IND(NUPT)=RVAR
GO TO 5
2225 CONTINUE
DIP(NUPT)=RVAR
GO TO 5
2250 CONTINUE
GO TO (5010,5020,5,5040,5050,5,5,5080,5090,5100,5110,99999),ITEM
C
C WRITE THE REAL VARIABLE.
C
2300 CONTINUE
LOGVAL=RSCDMP(CALALL,
1 MIERSE,OMODE,NCPW,
2 RVAR,DBNAME,UNITFM,UNITFA,UNITNM,
3 PMPREP,PMES,PMORCN,
4 RNWTL,RSGFRM)
GO TO 5
C
C READ THE INTEGER VARIABLE.
C
3100 CONTINUE
LOGVAL=ISCLDR(IVAR,CALALL,
1 MIERSE,IMODE,NCPW,
2 DBNAME,.FALSE.,
3 MENUFL,MENUNM,NITEMS,ITEMS,
4 PMPREP,PMES,PMORCN,
5 RNWTL,INTFM,IDEF)
IF (.NOT.LOGVAL) GO TO 5
GO TO (5,5,6030,5,5,6060,6070,5,5,5,5,99999),ITEM
C
C EDIT THE INTEGER VARIABLE.
C
3200 CONTINUE
LOGVAL=ISGEDT(IVAR,CALALL,
1 MIERSE,NCPW,
2 DBNAME,
3 MENUFL,MENUNM,NITEMS,ITEMS,
4 PMPREP,PMES,PMORCN,
5 IDEF)
IF (.NOT.LOGVAL) GO TO 5

```

```

MWC05400
MWC05410
MWC05420
MWC05430
MWC05440
MWC05450
MWC05460
MWC05470
MWC05480
MWC05490
MWC05500
MWC05510
MWC05520
MWC05530
MWC05540
MWC05550
MWC05560
MWC05570
MWC05580
MWC05590
MWC05600
MWC05610
MWC05620
MWC05630
MWC05640
MWC05650
MWC05660
MWC05670
MWC05680
MWC05690
MWC05700
MWC05710
MWC05720
MWC05730
MWC05740
MWC05750
MWC05760
MWC05770
MWC05780
MWC05790
MWC05800
MWC05810
MWC05820
MWC05830
MWC05840
MWC05850
MWC05860
MWC05870
MWC05880

```



```

GO TO (5,5,6030,5,5,6060,6070,5,5,5,5,5,99999),ITEM
C
C WRITE THE INTEGER VARIABLE.
C
3300 CONTINUE
      LOGVAL=ISCMP(CALALL,
1      MTERSE,OMODE,NCPW,
2      IVAR,DBNAME,PHORGN,
3      PMPREP,PMT S,
4      RNRFTL,ISCFRM)
      GO TO 5
C
C ASSIGN RVAR TO THE NEW SHIP LBP.
C
5010 CONTINUE
      LBPNEW=RVAR
      GO TO 5
C
C ASSIGN RVAR TO THE NEW SHIP DRAFT.
C
5020 CONTINUE
      HNEW=RVAR
      GO TO 5
C
C ASSIGN RVAR TO THE NEW SHIP SHP.
C
5040 CONTINUE
      SHPNEW=RVAR
      GO TO 5
C
C ASSIGN RVAR TO THE NEW SHIP VSUS.
C
5050 CONTINUE
      VSUSNU=RVAR
      GO TO 5
C
C ASSIGN RVAR TO THE NEW SHIP DPROP.
C
5080 CONTINUE
      DPRNEW=RVAR
      GO TO 5
C
C ASSIGN RVAR TO THE NEW SHIP W(200).
C
5090 CONTINUE
      W200NU=RVAR
C
C ASSIGN RVAR TO THE NEW SHIP W(201).
C

```

```

MWC05890
MWC05900
MWC05910
MWC05920
MWC05930
MWC05940
MWC05950
MWC05960
MWC05970
MWC05980
MWC05990
MWC06000
MWC06010
MWC06020
MWC06030
MWC06040
MWC06050
MWC06060
MWC06070
MWC06080
MWC06090
MWC06100
MWC06110
MWC06120
MWC06130
MWC06140
MWC06150
MWC06160
MWC06170
MWC06180
MWC06190
MWC06200
MWC06210
MWC06220
MWC06230
MWC06240
MWC06250
MWC06260
MWC06270
MWC06280
MWC06290
MWC06300
MWC06310
MWC06320
MWC06330
MWC06340
MWC06350
MWC06360
MWC06370

```



```

5100 CONTINUE
      W201NU=RVAR
      GO TO 5
C
C ASSIGN RVAR TO THE NEW SHIP W(203).
5110 CONTINUE
      W203NU=RVAR
      GO TO 5
C
C ASSIGN IVAR TO THE NEW SHIP PPTYPE.
C
6030 CONTINUE
      PPTNEW=IVAR
      GO TO 5
C
C ASSIGN IVAR TO THE NEW SHIP NSHIP1.
C
6060 CONTINUE
      NSHIPW=IVAR
      GO TO 5
C
C ASSIGN IVAR TO THE NEW SHIP PRPTYPE.
C
6070 CONTINUE
      PRPTNW=IVAR
      GO TO 5
C
C ERROR IN READING DATA FOR CURVE FITTING. INFORM USER.
C
8000 CONTINUE
      CALL STRPAK(MESS,LMS,qHK ,57)YOU HAVE READ IN TOO MANY NUMBERS
      FOR THIS PAIR OF DATA.4
      CALL MESOUT(MESS)
      CALL SIRPAK(MESS,LMS,qHK ,57)IRE-SELECT THE CURVEPTS ITEM IN MENU
      1 INPUT AND TRY AGAIN.4
      CALL MESOUT(MESS)
      GO TO 9999
C
C RETURN TO CALLING PROGRAM.
C
9999 CONTINUE
      RETURN
      END
MWC06380
MWC06390
MWC06400
MWC06410
MWC06420
MWC06430
MWC06440
MWC06450
MWC06460
MWC06470
MWC06480
MWC06490
MWC06500
MWC06510
MWC06520
MWC06530
MWC06540
MWC06550
MWC06560
MWC06570
MWC06580
MWC06590
MWC06600
MWC06610
MWC06620
MWC06630
MWC06640
MWC06650
MWC06660
MWC06670
MWC06680
MWC06690
MWC06700
MWC06710
MWC06720
MWC06730
MWC06740
MWC06750
MWC06760
MWC06770
MWC06780
MWC06790
MWC06800

```



```

C-----MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM-----MWC00010
C SUBROUTINE MWCOMP-----MWC00020
C-----SUBPROGRAM DESCRIPTION-----MWC00030
C SUBROUTINE MWCOMP USES THE DATA FROM EXISTING SHIPS TO ESTIMATE
C EITHER W(200) OR W(201) AS A FUNCTION OF SHIP. W(203) IS ESTIMATE
C BASED ON A SET OF FIXED PARAMETRIC CURVES.
C THE STANDARD UNITS OF THIS PROGRAM ARE FEET, TONS AND KNOTS.
C-----SUBPROGRAM ASSUMPTIONS-----MWC00070
C THE MAXIMUM NUMBER OF PAIRS OF DATA POINTS TO BE FITTED IS 10.
C-----LABELED COMMON VARIABLES-----MWC00100
C LABELED COMMON MDNCPW HAS BEEN DEFINED IN SUBROUTINE MAINPG.
C LABELED COMMONS CRVPTS AND WTLAG HAVE BEEN DEFINED IN SUBROUTINE
C MWTPNT. LABELED COMMON NEWINF HAS BEEN DEFINED IN SUBROUTINE MWCHRT.
C LABELED COMMON COEFFS HAS BEEN DEFINED IN SUBROUTINE MWCOEF.
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----MWC00150
C DEX SIRPAK
C IMOVIC
C MI SOUT
C DEX LIBRARY
C NONE
C MODULI
C LINF11
C-----MWC00240
C LABELED COMMONS
C
C COMMON /MDNCPW/ NCPW
C COMMON /CRVPTS/ NPTS,IND(10),DEP(10)
C COMMON /NEWINF/ LBPN,IIN,PTIN,SHIPN,NSHIN,VSUSN,PRTN,DPRN,W200N,
C 1 W201N,W203N
C COMMON /WTLAG/ WTLAG
C COMMON /COEFFS/ C(2)
C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
C INTEGER NCPW,NPTS,WTLAG
C INTEGER PTIN,NSHIN,PRTN
C INTEGER MESS(16),LMS
C LOGICAL CALALL,LOCALL
C LOGICAL MTERSE,LOGVAL,LMOVEG
C REAL IND,DEP,C
C REAL LBPN,IIN,SHIPN,VSUSN,DPRN,W200N,W201N,W203N
C REAL W,W203S,W203P,W203B
C REAL F,FF,RPM
C
C VARIABLE DATA DEFINITIONS
C
C DATA LMS/16/

```



```

C
C BRANCH ACCORDING TO WHICH WEIGHT ITEM IS TO BE ESTIMATED.
C
C      GO TO (100,100,300),WFLAG
C
C FOR MACHINERY WEIGHT ITEM W(200) AND W(201), FIT A STRAIGHT LINE TO
C THE DATA POINTS PROVIDED. OBTAIN THE SLOPE AND Y-INTERCEPT COEFFI-
C CIENTS.
C
C      100 CONTINUE
C          DO 150 I=1,NPTS
C              WRITE (18,7000) I,IND(I),DEP(I)
C          150 CONTINUE
C          7000 FORMAT(1X,110,1X,113.6,1X,E13.6)
C          CALL LINFIT(NPTS,IND,DEP,C)
C
C ESTIMATE THE NEW SHIP WEIGHT.
C
C          W=C(1)*SHIPN + C(2)
C          IF (WFLAG.EQ.2) GO TO 200
C          W200N=W
C          GO TO 99999
C
C      200 CONTINUE
C          W201N=W
C          GO TO 99999
C
C FOR W(203), FIRST ESTIMATE THE PROPELLER DIAMETER IF NOT SPECIFIED.
C
C      300 CONTINUE
C          IF (DPRN.NE.0.) GO TO 320
C          IF (HN.EQ.0.) GO TO 8100
C          IF (NSHN.EQ.1) GO TO 310
C          DPRN=.428*(HN*.04283)
C          GO TO 320
C
C      310 CONTINUE
C          DPRN=2.603*(HN*.629)
C
C ESTIMATE RPM.
C
C      320 CONTINUE
C          IF (VSUSN.EQ.0.) GO TO 8200
C          RPMN=96.12*(VSUSN/DPRN) + 52.15
C
C ESTIMATE WEIGHT FOR SHAFTING FOR FIXED PITCH AND GRP PROPELLERS
C DEPENDING ON TYPE OF PROPULSION PLANT.
C
C          IF (PPIN.LE.4) F=.36
C          IF (PPIN.GT.4) F=.20
C          IF (PRIN.EQ.2) GO TO 330

```

MWC00500
MWC00510
MWC00520
MWC00530
MWC00540
MWC00550
MWC00560
MWC00570
MWC00580
MWC00590
MWC00600
MWC00610
MWC00620
MWC00630
MWC00640
MWC00650
MWC00660
MWC00670
MWC00680
MWC00690
MWC00700
MWC00710
MWC00720
MWC00730
MWC00740
MWC00750
MWC00760
MWC00770
MWC00780
MWC00790
MWC00800
MWC00810
MWC00820
MWC00830
MWC00840
MWC00850
MWC00860
MWC00870
MWC00880
MWC00890
MWC00900
MWC00910
MWC00920
MWC00930
MWC00940
MWC00950
MWC00960
MWC00970
MWC00980


```

W203S=F*LBPN*NSIN*((.0134*((SHIPN/(NSIN*RPMN))**.6667))**.9497)
GO TO 340
330 CONTINUE
FI=F*LBPN*NSIN*(1.+ .5*FI0A1(NSIN))
W203S=FF*((.0134*((SHIPN/(NSIN*RPMN))**.6667))**.9497)
C ESTIMATE PROPELLER WEIGHT FOR FP AND CRP TYPES.
C
340 CONTINUE
IF (PRIN.EQ.2) GO TO 345
W203P=.00146*(DPRN**3.279)*NSIN
GO TO 350
345 CONTINUE
W203P=.00314*(DPRN**3.128)*NSIN
C ESTIMATE BEARING WEIGHT.
C
350 CONTINUE
W203B=.15*(W203S+W203P)
C ESTIMATE TOTAL W(203)
C
W203=W203S+W203P+W203B
GO TO 9999
C ERROR MESSAGE. DRAFT OF NEW SHIP NOT SPECIFIED.
C
8100 CONTINUE
CALL SIRPAK(MESS,LMS,4HK ,55HIA VALUE FOR NEW SHIP DRAFT IS NEEDED
1D. RETURN TO INPUT4
CALL MESOUT(MESS)
CALL SIRPAK(MESS,LMS,4HK ,14HJAND ENTER IT.4)
CALL MESOUT(MESS)
GO TO 9999
C ERROR MESSAGE. VSUS OF NEW SHIP NOT SPECIFIED.
C
8200 CONTINUE
CALL SIRPAK(MESS,LMS,4HK ,55HIA VALUE FOR NEW SHIP SPEED IS NEEDED
1D. RETURN TO INPUT4
CALL MESOUT(MESS)
CALL SIRPAK(MESS,LMS,4HK ,14HJAND ENTER IT.4)
CALL MESOUT(MESS)
C RETURN TO CALLING PROGRAM
C
99999 CONTINUE
RETURN
END
MWC00990
MWC01000
MWC01010
MWC01020
MWC01030
MWC01040
MWC01050
MWC01060
MWC01070
MWC01080
MWC01090
MWC01100
MWC01110
MWC01120
MWC01130
MWC01140
MWC01150
MWC01160
MWC01170
MWC01180
MWC01190
MWC01200
MWC01210
MWC01220
MWC01230
MWC01240
MWC01250
MWC01260
MWC01270
MWC01280
MWC01290
MWC01300
MWC01310
MWC01320
MWC01330
MWC01340
MWC01350
MWC01360
MWC01370
MWC01380
MWC01390
MWC01400
MWC01410
MWC01420
MWC01430
MWC01440
MWC01450
MWC01460
MWC01470

```



```

C-----MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM-----MW000010
C  SUBROUTINE OUTPUT(CALALL, IOLAG)MW000020
C-----SUBPROGRAM DESCRIPTION-----MW000030
C  SUBROUTINE OUTPUT PROVIDES THE USER WITH A MENU FROM WHICH TO
C  CHOOSE WHICH MODULE SEGMENT IT IS DESIRED TO OPERATE NEXT. THE
C  CHOICES ARE:MW000040
C  ALL MODULE OUTPUT VARIABLES MW000050
C  THE MODULE UNITS TO BE USED DURING INPUT AND OUTPUT MW000060
C  THE ESTIMATED MACHINERY WEIGHTS MW000070
C  THE COEFFICIENTS FROM LINEFITTING MW000080
C  THE UNITS MODULE ALLOWS THE USER TO SPECIFY THE LENGTH, FORCE
C  AND TIME UNITS TO BE USED DURING INPUT AND OUTPUT. THE LENGTH UNITS
C  ARE USED FOR LBP, PROPELLER DIAMETER, DRAGL AND POSSIBLY SPEED. THE
C  FORCE UNITS ARE USED FOR WEIGHTS AND THE TIME UNITS MAY BE USED FOR
C  SPEED. THE USER HAS THE OPTION OF USING 'KNOTS' FOR SPEED
C  EVEN IF 'NAUTICAL MILES' AND 'HOUR' ARE NOT THE LENGTH AND TIME
C  UNITS RESPECTIVELY FOR I/O. MW000090
C  THE ESTIMATED MACHINERY WEIGHTS ARE PROVIDED BY SUBROUTINE MWCHRT MW000100
C  WHICH INCLUDES OTHER SHIP CHARACTERISTICS AS WELL. MW000110
C-----SUBPROGRAM ASSUMPTIONS-----MW000120
C  NONE YET. MW000130
C-----OUTPUT VARIABLES-----MW000140
C  CALALL: .TRUE. IF THE INPUT VALUE OF CALALL WAS .TRUE. AND NO ERROR
C  OCCURRED WHEN READING OR EDITING AN ESSENTIAL VALUE. MW000150
C  .FALSE. IF THE INPUT VALUE OF CALALL WAS .FALSE. OR AN ERROR
C  OCCURRED WHEN READING OR EDITING AN ESSENTIAL VALUE. MW000160
C-----INPUT VARIABLES-----MW000170
C  CALALL: .TRUE. THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C  .FALSE. THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVE MW000180
C  IOLAG: DENOTES THE OPERATION TO BE PERFORMED MW000190
C  1 IF THE USER WISHES TO READ THE VARIABLES
C  2 EDIT
C  3 WRITE MW000200
C-----LABELED COMMON VARIABLES-----MW000210
C  LABLBD COMMON DIALG AND MDNCPW HAVE BEEN DEFINED IN SUBROUTINE
C  MAINPG. MW000220
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----MW000230
C  DEX MW000240
C  SIRPAK MW000250
C  LMOVIC MW000260
C  MINUTIN MW000270
C  DEX LIBRARY MW000280
C  NONE MW000290
C  MODULI MW000300
C  MVUNIT MW000310
C  MWCHRT MW000320
C  MWCOLF MW000330
C-----MW000340
C  MW000350
C  MW000360
C  MW000370
C  MW000380
C  MW000390
C  MW000400
C  MW000410
C  MW000420
C  MW000430
C  MW000440
C  MW000450
C  MW000460
C  MW000470
C-----MW000480
C  MW000490

```



```

C LABEL'D COMMONS
C
COMMON /DIALGF/ MTERSE
COMMON /MDNCPW/ NCPW

C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DEFINITIONS
C
INTEGER IOFLAG,NCPW,NUPT
INTEGER MENUM(2),NITEMS,ITEMS(10),ITEM
INTEGER MESS(16),IMS
INTEGER READ(2),EDIT(2),WRITE(2)
LOGICAL CALALL,LOCALL
LOGICAL MTERSE,LOGVAL,LMOVEC

C
C VARIABLE DATA DEFINITIONS
C
DATA LMS/16/
DATA MENUM/4HOUTP,4HUT /
DATA NITEMS/5/
DATA ITEMS/4HALL,4H ,
1 4HUNIT,4HS
2 4HWT,1,4HITEMS,
3 4HCOEF,4HFICT,
4 4HDONE,4H /
DATA READ /4HREAD,4H.< /
DATA EDIT /4HEDIT,4H.< /
DATA WRITE/4HWRITE,4H.</

C
C ACTIVATE THE LOCALL ALL OPTION IF THE CALLING PROGRAM REQUIRES IT.
C NOTE THAT IF LOCALL IS SET IN THIS MANNER, MENU 'OUTPUT' IS NOT
C DEFINED BY THE INVOCATION OF THIS SUBROUTINE.
C
LOCALL=CALALL
11 (LOCALL) GO TO 200

C
C PREPARE A PROMPTING MESSAGE FOR MENU 'OUTPUT' AND THEN PROVIDE THE
C MENU TO THE USER.
C
5 CONTINUE
IF (MTERSE) GO TO 40
CALL STRPAK(MESS,LMS,4H< ,4H)SELECT WHICH OUTPUT VARIABLE SEGMENT
11 TO 4
GO TO (10,20,30),IOFLAG
10 LOGVAL=LMOVEC(READ,1,6,NCPW,MESS,41,NCPW)
GO TO 50
20 LOGVAL=LMOVEC(EDIT,1,6,NCPW,MESS,41,NCPW)
GO TO 50
30 LOGVAL=LMOVEC(WRITE,1,7,NCPW,MESS,41,NCPW)
GO TO 50

```



```

40 CONTINUE
   CALL STRPAK(MESS,LMS,4HK ,22HWHICH OUTPUT SEGMENT?4)
50 CONTINUE
   ITEM=MENUN(MENUNM,NITEMS,ITEMS,MESS)
   GO TO (100,200,300,400,500),ITEM
C
C SET THE OUTPUT ALL OPTION.
C
100 CONTINUE
   LOCAL=.TRUE.
C
C READ, EDIT OR WRITE THE INPUT/OUTPUT MODULE UNITS.
C
200 CONTINUE
   CALL MWUNIT(LOCAL,IOFLAG)
   IF (LOCAL) GO TO 300
   IF (.NOT.CALALL) GO TO 5
   CALALL=.FALSE.
   GO TO 500
C
C CALL MWCHRT TO READ, EDIT OR WRITE THE ESTIMATED WEIGHT ITEM.
C
300 CONTINUE
   NUPT=0
   CALL MWCHRT(LOCAL,IOFLAG,NUPT),
   IF (LOCAL) GO TO 400
   IF (.NOT.CALALL) GO TO 5
   CALALL=.FALSE.
   GO TO 500
C
C CALL MWCOEF TO READ, EDIT OR WRITE THE LINE EQUATION COEFFICIENTS.
C
400 CONTINUE
   CALL MWCOEF(LOCAL,IOFLAG)
   IF (LOCAL) GO TO 500
   IF (.NOT.CALALL) GO TO 5
   CALALL=.FALSE.
C
C RETURN TO CALLING PROGRAM.
C
500 CONTINUE
   RETURN
   END
MW000990
MW001000
MW001010
MW001020
MW001030
MW001040
MW001050
MW001060
MW001070
MW001080
MW001090
MW001100
MW001110
MW001120
MW001130
MW001140
MW001150
MW001160
MW001170
MW001180
MW001190
MW001200
MW001210
MW001220
MW001230
MW001240
MW001250
MW001260
MW001270
MW001280
MW001290
MW001300
MW001310
MW001320
MW001330
MW001340
MW001350
MW001360
MW001370
MW001380
MW001390
MW001400
MW001410

```



```

C-----MACINERY WEIGHT ESTIMATING MODULE SUBPROGRAM-----MWC00010
C      SUBROUTINE MWCOFF(ALLFLG,IOFLAG)MWC00020
C-----SUBPROGRAM DESCRIPTION-----MWC00030
C      SUBROUTINE MWCOFF ALLOWS THE USER TO READ, EDIT OR WRITE THE
C      COEFFICIENTS OF THE EQUATION OF A STRAIGHT LINE.MWC00040
C-----SUBPROGRAM ASSUMPTIONS-----MWC00050
C      NONE YETMWC00060
C-----OUTPUT VARIABLES-----MWC00070
C      ALLFLG: .TRUE. IF THE INPUT VALUE OF ALLFLG WAS .TRUE. AND NO ERROR
C              OCCURRED WHEN READING OR EDITING A COEFFICIENTMWC00080
C              .FALSE. IF THE INPUT VALUE OF ALLFLG WAS .FALSE. OR AN ERROR
C              OCCURRED WHEN READING OR EDITING A COEFFICIENTMWC00090
C-----INPUT VARIABLES-----MWC00100
C      ALLFLG: .TRUE. IF THE ALL OPTION OF THE CALLING PROGRAM IS ACTIVE
C              .FALSE. IF THE ALL OPTION OF THE CALLING PROGRAM IS NOT ACTIVEMWC00110
C      IOFLAG: DENOTES THE OPERATION TO BE PERFORMEDMWC00120
C              =1 IF THE USER WISHES TO READ A COEFFICIENT
C              =2
C              =3
C              EDIT
C              WRITEMWC00130
C-----LABELLED COMMON VARIABLES-----MWC00140
C      LABELLED COMMON DIALGF,MDNCPW,INOUTF, AND REINOS HAVE BEEN DEFINED IN
C      SUBROUTINE MAINPG.MWC00150
C      C.....COEFFS INITIALIZED IN BLOCK DATA
C      C C : A TWO-ELEMENT ARRAY WHICH CONTAINS THE SLOPE AND Y-INTERCEPT
C              RESPECTIVELY OF AN EQUATION OF A STRAIGHT LINE
C      C.....COINIO INITIALIZED IN BLOCK DATA
C      C CNAME: THE DATABASE NAME OF THE ARRAY CONTAINING THE COEFFICIENTS
C              OF THE EQUATION OF A STRAIGHT LINE
C      C CFORM: THE DATABASE COMMENT OF THE COEFFICIENT ARRAY
C      C CFORM: THE FORMAT TO BE USED WHEN READING THE ARRAY FROM OR WRITING
C              IT TO A SEQUENTIAL FILE
C      C DEFC : AN ARRAY WHICH CONTAINS THE DEFAULT VALUES OF THE COEFFICIENTS
C      C NDEFC : THE NUMBER OF DEFAULT VALUES
C-----SUBPROGRAMS AND FUNCTIONS CALLED-----MWC00300
C      DEXMWC00310
C      NONE
C      DEX LIBRARYMWC00320
C      RATELRMWC00330
C      RAREDI
C      RARDMP
C      MODULL
C      NONE
C-----MWC00340
C      LABELLED COMMONSMWC00350
C      COMMON /DIALGF/ MIERSE
C      COMMON /MDNCPW/ NCPW
C      COMMON /INOUTF/ IMODE,OMODE

```



```

COMMON /REFNOS/ RNRFIL, RNWFIL
COMMON /COLFIS/ C(2)
COMMON /COINIO/ CFNAME, CFCMNT, COFORM, DEFC(2), NDEFC

C
C VARIABLE AND FUNCTION TYPE DEFINITIONS AND DIMENSIONS
C
  INTEGER IOFLAG, IMODE, OMODE, NCPW, RNRFIL, RNWFIL
  INTEGER CFNAME(2), CFCMNT(16), COFORM(2), PMES(16), UNITNM(3)
  INTEGER EFROM, NIO, MXTOGT, CGOI, CIROM
  LOGICAL ALLFLG, MIFRSE, LOGVAL
  LOGICAL RAILDR, RARED1, RARDMP
  REAL C, UNITFM, UNITFA, DEFC

C
C VARIABLE DATA DEFINITIONS
C
  DATA UNITNM/4HNONE,4H /
  DATA CFROM,NIO,MXTOGT,CGOI /1,2,2,2,2/
  DATA UNITFM,UNITFA /1.0,0.0/

C
C BRANCH ACCORDING TO THE OPERATION TO BE PERFORMED
C
  GO TO (100,200,300), IOFLAG

C
C READ THE COEFFICIENTS.
C
  100 CONTINUE
  LOGVAL=RAILDR(C,ALLFLG,CGOI,
  1 MIFRSE,IMODE,NCPW,
  2 CFNAME,MXTOGT,UNITFM,UNITFA,UNITNM,.TRUE.,
  3 .TRUE.,PMES,CFCMNT,
  4 RNRFIL,COFORM,
  5 NDEFC,DEFC)
  GO TO 99999

C
C EDIT COEFFICIENTS.
C
  200 CONTINUE
  LOGVAL=RARED1(C,ALLFLG,
  1 CFNAME,EFROM,NIO,
  2 UNITFM,UNITFA,UNITNM,.TRUE.,
  3 .TRUE.,PMES,CFCMNT,
  4 RNRFIL,COFORM,
  5 NDEFC,DEFC)
  GO TO 99999

C
C WRITE COEFFICIENTS.
C
  300 CONTINUE
  LOGVAL=RARDMP(ALLFLG,

```

```

MWC00500
MWC00510
MWC00520
MWC00530
MWC00540
MWC00550
MWC00560
MWC00570
MWC00580
MWC00590
MWC00600
MWC00610
MWC00620
MWC00630
MWC00640
MWC00650
MWC00660
MWC00670
MWC00680
MWC00690
MWC00700
MWC00710
MWC00720
MWC00730
MWC00740
MWC00750
MWC00760
MWC00770
MWC00780
MWC00790
MWC00800
MWC00810
MWC00820
MWC00830
MWC00840
MWC00850
MWC00860
MWC00870
MWC00880
MWC00890
MWC00900
MWC00910
MWC00920
MWC00930
MWC00940
MWC00950
MWC00960
MWC00970
MWC00980

```



```

MWC00990
MWC01000
MWC01010
MWC01020
MWC01030
MWC01040
MWC01050
MWC01060
MWC01070
MWC01080
MWC01090

```

```

MTERSE, OMODE, NCPW,
C, CNAME, CTROM, CCOI,
UNITM, UNIT1A, UNIT1N,
.TRUE., PMES, CCMNT,
RWFFIL, COFORM)

```

```

C
C RETURN TO CALLING PROGRAM.
C
99999 CONTINUE
      RETURN
      END

```



```

C..... SUBROUTINE TUNIT.....
COMMON /TUNITS/ PSTUN,UTOION
COMMON /TUNEO/ DBFUN,DBGUNC,TUNERM,DETFUN
INTEGER PSTUN,UTOION
INTEGER DBFUN(2),DBGUNC(16),TUNERM(2),DETFUN
DATA PSTUN/h/
DATA UTOION/h/
DATA DBFUN/hhiof,hhun /
DATA DBGUNC/hhiorc,hhe u,hhni1,hhio b,
1 hh us,hhed b,hhurin,hhg in,
2 hhpo1,hhouip,hhouic,hh ,
3 hh ,hh ,hh /
DATA TUNERM/hh(10,hh) /
DATA DEFUN/h/
C..... SUBROUTINE MGCIR1.....
COMMON /VARM/ INTRM,RSCTRM
COMMON /INI02/ IBPNA,CMN12,DEF2
COMMON /INI05/ UNAME,CMN15,DEF5
COMMON /INI09/ PPINAM,CMN19,DEF19
COMMON /INI20/ SUPNAM,CMN20,DEF20
COMMON /INI21/ NSHNM,CMN21,DEF21
COMMON /INI24/ VSUSNM,CMN24,DEF24
COMMON /INI27/ PRPNAM,CMN27,DEF27
COMMON /INI28/ DPRNM,CMN28,DEF28
COMMON /INI200/ W200NM,CM1200,DEF200
COMMON /INI201/ W201NM,CM1201,DEF201
COMMON /INI203/ W203NM,CM1203,DEF203
COMMON /NLVINT/ IBPNM,HNLM,PPINLM,NSHNM,VSUSNM,PRPNLM,
1 DPRNM,W200NM,W201NM,W203NM
INTEGR INTRM(2),RSCTRM(2)
INTEGR IBPNA(2),UNAM(2),PPINAM(2),SUPNAM(2),NSHNM(2),
1 VSUSNM(2),PRPNAM(2),DPRNM(2),W200NM(2),W201NM(2),
2 W203NM(2)
INTEGR CMN12(16),CMN15(16),CMN19(16),CMN20(16),CMN21(16),
1 CMN24(16),CMN27(16),CMN28(16),CMN200(16),
2 CM1201(16),CM1203(16)
INTEGR PPINLM,NSHNM,VSUSNM,PRPNLM
INTEGR DEF19,DEF21,DEF27
REAL DEF2,DEF5,DEF20,DEF24,DEF28,DEF200,DEF201,DEF203
REAL IBPNM,HNLM,SUPNM,VSUSNM,DPRNM,W200NM,W201NM,W203NM
DATA INTRM/hh(10,hh) /
DATA RSCTRM/hh(19,hh2) /
DATA IBPNA/hhbp,hh /
DATA UNAM /hh,hh /
DATA PPINAM/hhppy,hhpe /
DATA SUPNAM/hhshp,hh /
DATA NSHNM/hhnsua,hh1 /
DATA VSUSNM/hhvsds,hh /
DATA PRPNAM/hhprpt,hhpy /

```

```

BL000500
BL000510
BL000520
BL000530
BL000540
BL000550
BL000560
BL000570
BL000580
BL000590
BL000600
BL000610
BL000620
BL000630
BL000640
BL000650
BL000660
BL000670
BL000680
BL000690
BL000700
BL000710
BL000720
BL000730
BL000740
BL000750
BL000760
BL000770
BL000780
BL000790
BL000800
BL000810
BL000820
BL000830
BL000840
BL000850
BL000860
BL000870
BL000880
BL000890
BL000900
BL000910
BL000920
BL000930
BL000940
BL000950
BL000960
BL000970
BL000980

```


DATA	DPRNM/4HDPRO,4HP	/		BLOO0990
DATA	W200NM/4HW200,4H	/		BLOO1000
DATA	W201NM/4HW201,4H	/		BLOO1010
DATA	W203NM/4HW203,4H	/		BLOO1020
DATA	CMNT2 /4HLENG,4HH B,4HETWE,4HLEN P,4HERPE,4HNDIC,4HULAR,			BLOO1030
1	4HS (?,4H????,4H????,4H????),4H<		4H	BLOO1040
2	4H	/		BLOO1050
DATA	CMNT5 /4HMOLD,4HED D,4HRAFT,4H TO ,4HKEEL,4H (??,4H????,			BLOO1060
1	4H????,4H??)<	4H	4H	BLOO1070
2	4H	/		BLOO1080
DATA	CMNT19/4HTYPE,4H OF ,4HPROP,4HULST,4HON P,4HILANT,4H(RLE,			BLOO1090
1	4HID)<	4H	4H	BLOO1100
2	4H	/		BLOO1110
DATA	CMNT20/4H101A,4H IN,4HSTAL,4HED ,4HSHAF,4HT HO,4HRSFP,			BLOO1120
1	4HOWER,4H (??,4H????,4H????,4H??)<	4H	4H	BLOO1130
2	4H	/		BLOO1140
DATA	CMNT21/4HNUMB,4HER O,4HF PR,4HOPEL,4HLER ,4HSHAF,4HIS<			BLOO1150
1	4H	4H	4H	BLOO1160
2	4H	/		BLOO1170
DATA	CMNT24/4HMAXI,4HUM ,4HCONT,4HINUO,4HIUS S,4HUSTA,4HINED,			BLOO1180
1	4H SPE,4HED (,4H????,4H????,4H????,4H)<	4H	4H	BLOO1190
2	4H	/		BLOO1200
DATA	CMNT27/4HTYPE,4H OF ,4HPROP,4HELLE,4HR (R,4HELD),4H<			BLOO1210
1	4H	4H	4H	BLOO1220
2	4H	/		BLOO1230
DATA	CMNT28/4HPROP,4HELLE,4HR DI,4HAMET,4HER (,4H????,4H????,			BLOO1240
1	4H????,4H)<	4H	4H	BLOO1250
2	4H	/		BLOO1260
DATA	CMT200/4HMI O,4HF BO,4HLER,4HS AN,4HD LN,4HERGY,4H CON,			BLOO1270
1	4HVERT,4HERS ,4H(???,4H????,4H????,4H??)<	4H	4H	BLOO1280
2	4H	/		BLOO1290
DATA	CMT201/4HMI O,4HF PR,4HOPEL,4HION,4H UNI,4HITS (,4H????,			BLOO1300
1	4H????,4H??)<	4H	4H	BLOO1310
2	4H	/		BLOO1320
DATA	CMT203/4HMI O,4HF PR,4HOPEL,4HLERS,4H, SH,4HAFT1,4HING A,			BLOO1330
1	4HND B,4HEARI,4HINGS ,4H(???,4H????,4H????,4H??)<			BLOO1340
2	4H	/		BLOO1350
DATA	DEF2,DEF5,DEF20,DEF24,DEF28 /0.0.0.0.0.0.0.0.0.0.0/			BLOO1360
DATA	DEF19,DEF21,DEF27 /0.0.0/			BLOO1370
DATA	DEF200,DEF201,DEF203 /0.0.0.0.0.0.0/			BLOO1380
DATA	LBPNEW,INCW,PPTNEW,SHIPNEW,NSHNEW,VSSUSNU,PRITNEW,DPRNEW,			BLOO1390
1	W200NU,W201NU,W203NU			BLOO1400
2	/420.,15.1,2.7000.,2.33.,1.12.,229.3,128.2,127.6/			BLOO1410
C.....SUBROUTINE INPUT.....				BLOO1420
COMMON /WFLAG/ WFLAG				BLOO1430
COMMON /CRVPTS/ NPTS, IND(10), DEP(10)				BLOO1440
INTEGER WFLAG, NPTS				BLOO1450
REAL IND, DEP				BLOO1460
DATA WFLAG /1/				BLOO1470


```

DATA NPIS/0/
DATA IND(1),IND(2),IND(3),IND(4),IND(5),IND(6),IND(7),
1 IND(8),IND(9),IND(10)
2 /0.,0.,0.,0.,0.,0.,0.,0.,0.,0./
DATA DEP(1),DEP(2),DEP(3),DEP(4),DEP(5),DEP(6),DEP(7),
1 DEP(8),DEP(9),DEP(10)
2 /0.,0.,0.,0.,0.,0.,0.,0.,0.,0./
C.....SUBROUTINE MWCOEF.....
COMMON /COIN10/ CFNAME,CFGMNT,COFORM,DEFC(2),NDEFC
INTEGER CFNAME(2),CFGMNT(16),COFORM(2),NDEFC
REAL C,DEFC
DATA C /0.,0./
DATA CFNAME/HHSL C,4HHOEF/
DATA CFGMNT/HHCOEF,4HFCI,4HIENTS,4H OF ,4HITHE ,4HIFQUA,4HITION,
1 4H OT ,4HIA ST,4HIRATG,4HHIT L,4HINE ,4H(SLO,4HPE,Y,
2 4H-INI,4HICP
DATA COFORM/HH2(E1,4H3.6)/
DATA NDEFC,DEFC(1),DEFC(2) /2,0.00234,48.09/
END

```

```

BLO01480
BLO01490
BLO01500
BLO01510
BLO01520
BLO01530
BLO01540
BLO01550
BLO01560
BLO01570
BLO01580
BLO01590
BLO01600
BLO01610
BLO01620
BLO01630
BLO01640
BLO01650
BLO01660
BLO01670

```



```

C-----MACHINERY WEIGHT ESTIMATING MODULE SUBPROGRAM-----LIN00010
C      SUBROUTINE LINIT(N,X,Y,C)LIN00020
C-----SUBPROGRAM DESCRIPTION-----LIN00030
C      SUBROUTINE LINIT CALCULATES THE COEFFICIENTS FOR A STRAIGHTLIN00040
C LINE FIT OF DATA POINTS. THE COEFFICIENTS A AND B AND X AND Y ARELIN00050
C DEFINED AS:LIN00060
C      Y=AX+B LIN00070
C      THE VALUES OF A AND B ARE RETURNED TO THE CALLING PROGRAM AS THELIN00080
C FIRST AND SECOND ELEMENTS RESPECTIVELY OF ARRAY C. LIN00090
C-----SUBPROGRAM ASSUMPTIONS-----LIN00100
C      NONE YET LIN00110
C-----OUTPUT VARIABLES-----LIN00120
C      C : AN ARRAY CONTAINING THE VALUES OF THE SLOPE (A) AND THELIN00130
C      Y-INTERCEPT (B) OF THE LINE LIN00140
C-----INPUT VARIABLES-----LIN00150
C      N : THE NUMBER OF PAIRS OF DATA POINTS LIN00160
C      X : AN ARRAY CONTAINING THE ABSCISSAS OF THE DATA POINTS LIN00170
C      Y : AN ARRAY CONTAINING THE ORDINATES OF THE DATA POINTS LIN00180
C-----LIN00190
C      C VARIABLE DEFINITIONS AND DIMENSIONS
C
C      INTEGER N
C      REAL X(N),Y(N),C(2)
C      REAL A,B,SUMX1,SUMX12,SUMY1,SUMY12,SMXY1
C      REAL R2,SCMAXY,STGMAY
C
C      INITIALIZE VARIABLES.
C
C      SUMX1=0.
C      SUMX12=0.
C      SUMY1=0.
C      SUMY12=0.
C      SMXY1=0.
C
C      CALCULATE THE COEFFICIENTS A AND B.
C
C      DO 10 I=1,N
C      SUMX1=SUMX1+X(I)
C      SUMX12=SUMX12 + X(I)*X(I)
C      SUMY1=SUMY1 + Y(I)
C      SUMY12=SUMY12 + Y(I)*Y(I)
C      SMXY1=SMXY1 + X(I)*Y(I)
C 10 CONTINUE
C      B ((SUMX12*SUMY1)/SUMX1 - SMXY1)/((N*SUMX12/SUMX1) - SUMX1)
C      C(2)=B
C      A (SUMY1-N*B)/SUMX1
C      C(1)=A
C

```



```

C CALCULATE AND PRINT THE GOODNESS OF FIT.
C
      SGMXY=SMXI*YI-((SUMXI*SUMYI)/N)
      SIGMAX=SUMXI2-((SUMXI2)/N)
      SIGMAY=SUMYI2-((SUMYI2)/N)
      R2=(SGMXY2)/(SIGMAX*SIGMAY)
      WRITE(18,99) R2
      99 FORMAT(3X,'A STRAIGHT LINE WAS FIT TO THE DATA WITH A GOODNESS OF
         1 FIT OF ',1X,16.4)
C
C RETURN TO CALLING PROGRAM
C
      RETURN
      END

```

LIN00500
 LIN00510
 LIN00520
 LIN00530
 LIN00540
 LIN00550
 LIN00560
 LIN00570
 LIN00580
 LIN00590
 LIN00600
 LIN00610
 LIN00620
 LIN00630

Thesis
C338367
c.1

CeLotto

195519

An investigation
into the use of data
bases in computer-
aided naval ship de-
sign.

Thesis
C338367
c.1

CeLotto

195519

An investigation
into the use of data
bases in computer-
aided naval ship de-
sign.

thesC338367

An investigation into the use of data ba



3 2768 002 09129 0

DUDLEY KNOX LIBRARY